universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

The Preservation of Software-Based Art

verfasst von | submitted by

Yusra Dellali BA

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Arts (MA)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 647
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Digital Humanities
Betreut von | Supervisor:	Mag. Dr. Thomas J.J. Wallnig Privatdoz. MAS

i. Abstract

This master's thesis explores the various methods and processes for the long-term preservation of software-based art. While the field of new media art has existed for decades, the preservation of such artworks remains less common and advanced as one might expect. The aim of this work is to illuminate and compare the various preservation techniques, providing a concise overview of the status quo.
This research is conducted in two stages: first, the basis is a thorough review of contemporary literature, outlining the various preparatory aspects for successful preservation. It also explains and compares the different conservation methods.
A central point here is the importance of preserving source code, which has been historically neglected in the art world but has gained increasing recognition in recent years.
This is followed by a brief discourse on the institutional field of the Ars Electronica, and the eponymous Art Festival hosted in the Upper-Austrian city of Linz, highlighting a concise explanation of its most important areas.
The second stage comprises three case studies of selected software-based artworks. These three case studies are contextually classified, described, and then analysed using the previously established theoretical methods and critically negotiated regarding their probability of preservation.
The resulting findings provide valuable insights and a highly topical overview of the current state of existing methods for preserving software-based art. Furthermore, they create forum for a more efficient approach, thereby ensuring the success of long-term preservation.

ii. Abstract (Deutsch)

Diese Masterarbeit befasst sich mit den verschiedenen Methoden und Prozessen zur Langzeiterhaltung von Software-basierter Kunst. Das Feld neuer Medienkunst existiert nun schon seit Jahrzehnten, wobei die Erhaltung der Kunstwerke in diesem Feld noch nicht so gängig und weit fortgeschritten ist, wie man annehmen würde. Ziel dieser Arbeit ist es, die verschiedenen Techniken zu beleuchten und zu vergleichen, um so ein möglichst aktuelles Resümee über die Lage ziehen zu können.
Vorgegangen wird dabei in zwei Schritten: die Basis bildet eine gründliche Durchschau aktueller Literatur, welche sich mit den verschiedenen vorbereitenden Aspekten für eine erfolgreiche Erhaltung auseinandersetzt und die unterschiedlichen Erhaltungsmethoden erläutert und vergleicht. Einen zentralen Punkt stellt dabei die Wichtigkeit der Erhaltung von Source Code dar, welcher in der Vergangenheit im Kunstbereich vernachlässigt wurde, inzwischen aber immer mehr an Aufmerksamkeit gewinnt. Anschließend folgt ein kurzer Diskurs in das institutionelle Feld der Kunsteinrichtung der Ars Electronica, und dem gleichnamigen Kunstfestival der oberösterreichischen Stadt Linz, mit einer prägnanten Erläuterung ihrer wichtigsten Bereiche.
Den zweiten, darauffolgenden Schritt bilden drei Case Studies ausgewählter Software-basierter Kunstwerke. Diese werden kontextuell eingeordnet, beschrieben und anschließend anhand der zuvor theoretisch etablierten Methoden analysiert und in Hinblick auf ihre Erhaltungswahrscheinlichkeit kritisch ausgehandelt.
Die daraus entstehenden Ergebnisse liefern wertvolle Einblicke und eine höchst aktuelle Übersicht über den momentanen Stand existenter Methoden zur Erhaltung Software-basierter Kunst. Des Weiteren öffnen sie den Raum für eine effizientere Herangehensweise zum Zweck einer erfolgreicheren, langjährigen Erhaltung.

Table of Contents

1 Method And Structure of the Thesis

1.1 Position of the Problem

Since childhood, I remember our father teaching us about the importance of saving data and performing regular backups on our laptop or PC. In line with this lies the fundamental practice of this thesis: data preservation. - But not just any data. This thesis focuses specifically on the preservation of software-based art (files).
The topic of this thesis will be the preservation of software-based art illustrated using practical examples found online and at the Arts Electronica Festival.
For over 40 years, Linz has been a pioneer in modern, digital museum landscape, in contrast to other conservative scope of museums in Austria. The frontrunner of this movement is the institution of the Ars Electronica, along with its eponymous festival, known as the Ars Electronica Festival, occurring annually at the beginning of September. Every year, over 1000 artists, scientists, developers, entrepreneurs, and activists globally travel to Linz, showcasing an average of 2700 artworks.
Although contemporary artists have been utilising computers and software since the mid-20th century, the number of preserved artworks remains relatively small – even within major international collections with extensive time-based media holdings. However, this is subject to change: over the last decade, artists have continued to embrace the newest digital technologies in their works, while collectors are growing increasingly confident in their ability to care for these works (Engel & Phillips, 2023, p. 453). Despite their difference in experience, software-based artworks have many similarities with other types of time-based media[1]: they must be treated and understood as dynamic systems composed of interdependent software and hardware components, with their industrially produced technologies being variable and/or replaceable. Their inherent and inevitable change must be managed carefully to prevent damaging their integrity (Engel & Phillips, 2023, p. 454). The extreme system complexity they display, coupled with faster technological obsolescence and failure, makes their preservation more challenging than caring for "traditional time-based media works" (Laurenson, 2014 in: Engel & Phillips, 2023, p. 454), making expertise required for their maintenance is highly specific.

[1] "Usually time-based media are video, slide, film, audio or computer based. Part of what it means to experience the art is to watch it unfold over time according to the temporal logic of the medium as it is played back" (Tate, 2021).

Institutions, such as Tate Museum in London and the Solomon R. Guggenheim Museum in New York have been archiving software-based artworks for over a decade. However, Ars Electronica lacks in this regard, as software-based artworks presented at the Ars Electronica Festival are not fully archived and preserved. They are only documented in the festival's print catalogue or in simple database entries with corresponding text explanations, amplified with photographs, video recordings, or links to the artist's homepages.

However, what is digital preservation? The Digital Preservation Coalition Handbook defines digital preservation as

> […] the series of managed activities necessary to ensure continued access to digital materials for as long as necessary […] and all of the actions required to maintain access to digital materials beyond the limits of media failure or technological and organisational change (Digital Preservation Coalition, 2025a).

Why is digital art preserved? Without proper attention, digital information can be lost, preventing its future value from ever being realised. One example of this is "digital assets [that] might be cultural - data that enriches our lives" (Digital Preservation Coalition, 2025b) - software-based artworks in this thesis.

Before an institution can advance and preserve an artwork, it must first establish a preservation plan. This is essential for understanding why the artwork is being preserved in a particular setting. Patrícia Falcão recommends considering a few questions to determine the reason for preserving artwork within an institution's particular setting.

The first question she proposes is how the institution intends to use the artwork. To answer this, the institution must determine if the work is part of the artist's private archive for research purposes or part of a collection intended for loan and display. If the latter is the case, the institution might decide whether the work will be used as part of a teaching collection or to lay the focus on technologies of production next to the artwork itself. Other important considerations include whether the artwork would be on permanent or temporary display or if it is a commissioned piece and what will happen to the artwork after its exhibition (Falcão, 2024, p. 3).

While these questions might initially seem challenging, they are important for identifying the most appropriate preservation strategies as well as considering the available resources and infrastructure.

Currently, three categories of digital media preservation are considered to be in a vulnerable or critically endangered state:

1. Recently commissioned or completed media art.
2. Exhibition content.
3. Manuals, documentation and associated materials. This judgement was made by the Digital Preservation Coalition in 2024 (Digital Preservation Coalition, 2024c, 2024a, 2024b).

Vulnerable means "[…] the technical challenges to preservation are modest but responsibility for care is poorly understood, or […] the responsible agencies are not meeting preservation needs" (Digital Preservation Coalition, 2024d). This is the case for recently commissioned or completed media art – media art currently being displayed in a gallery or in the process of being displayed (Digital Preservation Coalition, 2024c). The 2024 DPC Jury agreed the classification remains *Vulnerable* but "[…] with a trend towards greater risk because the imminence of action is time-sensitive, requiring working with the artist to get the documentation from them about their work and what is needed before it is too late" (Digital Preservation Coalition, 2024c). The jury further reported that the loss of such artworks would significantly impact many people and sectors, and only a minimal effort would be required to preserve the materials in this group.

Both exhibition content and manuals, documentation, and associated materials are categorised as *critically endangered* – "[…] they face material technical challenges to preservation, there are no agencies responsible for them or those agencies are unwilling or unable to meet preservation needs" (Digital Preservation Coalition, 2024d).

Exhibition content refers to born-digital or hybrid-digital content created and/or commissioned for exhibitions, but has not been transferred into a collection (Digital Preservation Coalition, 2024a). Examples include media art, digital artworks, and immersive works, among others. The 2024 Council maintained the current risk classification for these materials but identified three main emerging and broader risks: cybersecurity threats, machine leaning/artificial intelligence (ML/AI) and political factors.

Examples of manuals, documentation, and associated materials are "Manuals created for the object and/or from the supplier or manufacturer; conservation records and other forms of documentation" (Digital Preservation Coalition, 2024b).

Compared to recently commissioned or completed media art, the loss of exhibition content and manuals will negatively impact many people and sectors. The DPC wrote

and warned in 2024 that the likelihood of loss is high, as necessary tool for preservation may arrive late, rendering materials irretrievable and lost.
This underscores the critical need for a meticulous approach regarding the preservation of software-based artworks. Due to its hybrid nature, digital art is inherently more fragile than other digital objects, and the only way to guarantee its continuance is through systematic collection and preservation.

1.2 Method and Structure

The objective of this thesis is to examine the preservation of software-based art, emphasising on the overlooked aspect of art preservation, as highlighted in the preceding outline of the problem.
The research will be undertaken in several chapters.
The first chapter establishes the context by providing a brief, explanatory introduction of the challenges associated with preserving software-based art.
The second chapter situates software-based art within the broader sphere of new media art. This will be achieved by providing a brief overview of the history of new media and software-based art, as well as how the two artistic currents closely grew together and began to overlap. The chapter concludes with a comparison of Ars Electronica with similar institutions (Tate Modern, MoMA, Guggenheim, ZKM) and highlighting various significant past and ongoing preservation projects.
The third chapter provides a comprehensive examination of software-based art preservation. It begins with an introductory explanation of the topic in general, which includes outlining methods for engaging with, understanding and reviewing artwork. After explaining general risk assessment, this leads into an analysis of various components of an artwork. Furthermore, the discussion then extends to introducing software and source code, thereby leading to exploring programming languages, platforms and their significance regarding preservation. The last section concludes by discussing the general treatment and preservation strategies for software-based artworks, illustrating different environment intervention strategies and code intervention.
The fourth chapter specifically addresses the Ars Electronica, offering a brief overview of their history. It then examines its museum and Futurelab, followed by an outline of its archive and the associated database ecosystem. The chapter ends with a discussion of the famous Ars Electronica Festival and its primary art award, the Prix Ars Electronica.

The fifth chapter presents three case studies analyses. These analyses contain a brief introduction, an overview of the artist(s), past exhibitions of the artwork, and a statement of significance, split into artistic/historical significance and technical/research significance. Following this, the next two chapters examine risk assessment, long-term risks, recoverability and preservation strategies. In addition, the specific file formats of each artwork are scrutinised, along with a brief digression about software maintenance, source code annotation and source code preservation.
The sixth chapter concludes the thesis, by summarising the subject matter and providing an outlook on the future of software-based art preservation.
Finally, the seventh, eighth and ninth chapters comprise the bibliography, list of tables and list of figures, respectively.

2 New Media Art – Software-Based Art

"The complexity of the system is created by the number of elements, its array of functions, the mix of bespoke and commercial software, and ultimately its ambition" (Laurenson, 2014, p.78).
This chapter provides a brief overview of software-based art within the broader field of new media art, examining how institutions have historically and currently approached the preservation of art works.
New media art or time-based media art will be used interchangeably in the following to refer to artworks "[…] whose primary medium is video, film, 35 mm transparencies, or audio" (Laurenson, 2014, p.73). Furthermore, it can be broadly defined as a term encompassing any art form that utilises new media technologies, particularly those technologies developed since the late 20th century. These includes digital art, video art, interactive installations and bio art (Grau, 2016).
The systems of which software-based art are made of are less well-defined and more diffuse, based on the simple fact that they operate in a "[…] more diverse, non-standardized, and fast-moving technological environment" (Laurenson, 2014, p.73).
Software-based art plays a crucial role in new media art, explicitly focusing on art pieces where computer software plays a central role in the creation, manipulation, or presentation of the artwork. It further highlights the use of code, algorithms, and digital tools as artistic mediums, allowing artists to explore new forms of expression and creatively engage with technology.
This leads to a significant overlap between software-based art and other new media art forms – modern interactive installations often heavily rely on software, with digital art

already being inherently software-based (The Solomon R. Guggenheim Foundation, 2020). Software-based art often serves as a foundational element within many new media artworks, thereby providing the structural and functional basis for interactive experiences, digital simulations and generative art.

Its significance within the new media art landscape is undeniable – software-based art is continuously pushing the boundaries of new media art by introducing new aesthetic possibilities through digital manipulation and computational processes (Aris et al., 2023), enabling complex interactive experiences and dynamic art pieces to respond to viewers or their environments (Liakopoulos, 2023) while also raising important questions about the nature of art, creativity, and authorship in the digital age (centerforartlaw, 2024).

Software-based art preservation has brought together experts from all over the world – popular preservation institutions in the English-speaking realm range from the Tate Modern in London, to the Museum of Modern Art and the Guggenheim in New York to the ZKM in Karlsruhe (Germany) and the Ars Electronica in Linz (Austria) in German-speaking countries. All these institutions are actively working to establish guidelines, consensus and best practices to effectively manage the technology-based work of arts within their respective museum or archive walls.

The aforementioned institutions will be explicitly explored below, thereby illustrating the necessary care and specific infrastructure required to successfully preserve software-based art.

The Tate Modern has been collecting software-based artworks since the early 2000s, and with the same duration, software-based artworks have challenged "[…] existing modes of conservation through their contingencies on bespoke software programs, complex systems of interconnected components and a rapidly changing technological landscape that tends to accelerate processes of obsolescence" (Ensom et al., 2022). In response to this, Tate's Time-based Media and Collection Care Research teams have been participating in a project of collective research, "Software-based Art Preservation", over the past decade. Project outcomes have been a Software-based Artwork Conservation Report Template, a Disk Imaging Guide, a Disk Imaging Report Template, an Emulation Report Template and Metadata Fields for Computer and Software Components (Ensom et al., 2022).

Similar to the Tate Modern, the Museum of Modern Art in New York has taken a similar approach through Matters in Media Art, a "[…] multi-phase project designed to provide guidelines for care of time-based media works of art (e.g., video, film, audio

and computer based installations)" (MoMa, 2023). Matters in Media Art emerged in 2003 by various conservators, curators, media technical managers and registrars from the MoMa, SFMOMA, New Art Trust and Tate (MoMa, 2023; Tate, 2022).
The Guggenheim has taken a very similar approach – they introduced the "Conserving Computer-Based Art Initiative" (CCBA), an "[…] ongoing, practice-based research project that focuses on the preservation needs of computer-based artworks in the Guggenheim's collection" (The Solomon R. Guggenheim Foundation, 2020). The project involves the collaboration of the museum's CCBA Fellowship, the academic-museum partnership between New York University's Department of Computer Science and the Guggenheim's Conservation Department alongside museum's fellows, interns and conservation staff (The Solomon R. Guggenheim Foundation, 2020).
The Zentrum für Kunst und Medien Karlsruhe has been a pioneer in digital preservation within the German-speaking countries. Their team of specialists has extensive knowledge in restoration, electrical engineering, informatics and art history; the restoration team from their Knowledge Department and the Museum and Exhibiting Technology department have been documenting and restoring works of art since 2004 (Serexhe, n.d.; ZKM Karlsruhe, n.d.-b). They are also working on the continuous development of "[…] standardised methods of documentation and long-term preservation of electronic or digital works of art" (ZKM Karlsruhe, n.d.-b), thereby promoting research projects and giving interns and scholarship holders opportunity to obtain access and insight about their current work and state of the art research (ZKM Karlsruhe, n.d.-b). The restoration team works closely with the Laboratory for Antiquated Video Systems, one of the world's leading laboratories for video restoration, which is assigned to the Knowledge Department (ZKM Karlsruhe, n.d.-a).
In Austria, the Ars Electronica in Linz has been a driving force in digital media art since its first festival in 1979 (Ars Electronica Linz GmbH & Co KG, n.d.-b). They hold one of the world's most extensive archives on digital media art, "[…] spanning the years since 1979, including documentation on the Ars Electronica Festival, the Prix Ars Electronica archive with artists' submissions, as well as documentation on activities of the Ars Electronica Futurelab, the Ars Electronica Center and all other areas of Ars Electronica" (Ars Electronica Linz GmbH & Co KG, 2024b). They have been active in preservation and documentation of media since 2009 and have been digitising and expanding their archive since the 1990s (Ars Electronica Linz GmbH & Co KG, 2024a), with their latest project being the Ars Electronica Archive Relaunch (Ars Electronica Linz GmbH & Co KG, n.d.-a).

3 Preservation of Software-Based Art

After embedding software-based art in the bigger context of new media art, we will now proceed with the chapter from which this thesis got its name – the preservation of software-based art.

This chapter is split into eight sections: the first addresses the general first contact with a software-based artwork, while the second establishes a general understanding and review of a software-based artwork. In the third section, a risk assessment and crucial items will be examined closer, to then proceed to the different parts making up an artwork and their subsequent analysis. The fifth section introduces software and source code, different types of software and possible methods for analysis and documentation of software. Programming languages, platforms, and their significance regarding preservation are the topics of the sixth section, which lays the necessary groundwork for a general treatment and preservation strategies for software-based art in the seventh section. The eighth section consists of concluding remarks.

This chapter aims to lay a foundational basis for an understanding of the preservation of software-based art, which will be practically employed in Chapter 5.

3.1 First Contact with a Software-Based Artwork

The first encounter with a software-based artwork always involves an initial examination process. This is because first-hand observation of the artwork in its installed and running state is crucial for conservation personnel. This is essential to establish the artwork's documentary significance, understand its original structure and materials, and assess any deterioration, alterations, or loss. Documenting these findings is a key outcome of this first examination.

For software-based art, this first impression and documentation are particularly important in order to verify the artwork as a unified piece and to explore its different layers (Soares & Simão, 2020, p. 49). This first feel provides an immediate understanding "[…] of its intended behaviors and potential implications for conservation and maintenance and critically informs the acquisition process" (Engel & Phillips, 2023, p. 457). Complementary research about contextual or technical information during acquisition is also necessary due to the inherent variability of every software-based artwork.

Key information sources for this process include artist websites, catalogue texts, manuals, correspondences, media art databases (Rhizome, Archive of Digital Art), and recordings of the artwork. Documenting and analysing artwork components is an

iterative process, further informing necessary understanding, which will be explained in more detail in subchapter 4. Ultimately, as Engel and Phillips note, a software-based artwork's object file will contain comprehensive documentation, including inventories, instructions, risk assessments, source code documentation, and metadata (Engel & Phillips, 2023, p. 458).

3.2 Understanding and Reviewing the Artwork

To ensure appropriate care and preservation for a software-based artwork, conservation personnel "[…] need to identify its interconnected hardware and software components and understand their relative importance in realizing the artwork's work-defining properties, as well as how each component potentially affects the way the system behaves" (Engel & Phillips, 2023, p. 461). Following this, it is important to classify the artwork's connected parts – both hardware and software. Artists often provide materials like a computer with accessories ready for display and either a single program file or collections of files. Other supplies can include source code, additional exhibition hardware, sculptural pieces, hard drives with backups, added software, and spare parts (Engel & Phillips, 2023, p. 461). It is a good idea to use an inventory to keep track of artworks consisting of several different pieces.

This inventory should list basic details about the software, hardware, and data involved—explaining what each element does and how they all connect. For hardware, it should also note things like connectors and accessories and specify the relevant protocols, standards, models, and brands. A conservation report - similar to the one created by the Tate Modern in London (Tate, 2020) - can be helpful for this.

All of the aforementioned points must be considered carefully to keep the already artist-created, established dependency intact. Falcão et al. describe dependency as follows:

> We use the word dependency to describe the context under which change in one or more entities has an impact on other entities of the ecosystem of a work. A dependency is a connection, or relation, between different elements in a system; if one of the elements is missing or fails, the system will function differently or stop working altogether. (Falcão et al., 2016)

If this is applied to software-based artworks, resources to consider include software, hardware and data, but also "entities that perform activities on the system" (Falcão et al., 2016).

Prior to the preservation of a software-based artwork, a thorough inspection is necessary: this inspection serves to verify that all required deliverables are present, that

the artwork is operating as intended, and that documentation exists which details its behaviour within an artist-approved or artist-supplied context (Electronic Arts Intermix, n.d.-c).

To properly evaluate the artwork, a suitable inspection environment must be established, made of both appropriate hardware and software. In certain instances, this may involve only a standard desktop computer equipped with the designated operating system or a limited selection of web browsers. In other cases, a complete installation of the artwork may be required to make an adequate inspection possible.

Powering up a software-based artwork for the first time can represent a critical moment. Therefore, previously to powering up the system, it should be confirmed that a flawless and identical copy of all data and software is stored safely. Creating a backup is particularly important if the artwork is installed on a computer provided by the artist. In such cases, the optimal backup strategy involves generating a disk image of the entire computer's storage devices, including all installed applications and system software. Falcão summarises this very adequately:

> By creating an exact copy of the original disk we can start to investigate the software components without putting the original computer/s at risk, and we can also test the hardware while being assured that we can recover the system if it fails. (Falcão, 2019, p. 281)

Two excellent guides for this are the Disk Imaging Guide and the Disk Imaging Report templates which can be found on the website of the Software-based Art preservation project at Tate (Ensom, 2021; Time-based Media Conservation, Tate, 2020a).

Only after the successful creation of a disk image should the computer be powered on, with all appropriate hardware connected to ensure full functionality during testing and documentation procedures. Before operating the artwork itself, it is sensible to utilise the computer's integrated system utilities, such as the *System Report* function, to "create basic documentation of the hardware and software environment" (Engel & Phillips, 2023, p. 463). After the completion of this step, the software can be initiated.

At this phase in the preservation process, it is recommended to produce video documentation of the artwork and the user interface of the computer system while they are running.

Jarczyk et al. have elaborated further on why this is of utmost importance:

> The key point discussed was that video is essential to document not only the behaviour of these works, but also their installation processes. This can mean recording different hardware components being connected or screencasts demonstrating how to set-up an artwork's software. For example, if a work is being installed for the first time in an institution, video documentation of a previous installation process and a video of the work running are likely to be more helpful than a documentation folder. (Jarczyk et al., 2017, p. 13)

Conservation experts need (video) documentation to determine whether a) the components provided by the artist are complete and functional, b) whether any essential information for running the artwork is missing, c) which specific software (including version and configuration details) the artwork employs, d) which calibration procedures and set-up are necessary for the correct software execution, and, assuming correct operation of the artwork, e) the intended aesthetic and functional behaviours of the software (Engel & Phillips, 2023, p. 463).

Coupled with the aforementioned documentation, conservation experts are encouraged to employ the framework of "work-defining properties". This concept, initially proposed by Laurenson (2006), serves to outline the attributes of the artwork that are fundamental to its identity - attributes which "[…] are essential for its uncompromised integrity and should be preserved" (Engel & Phillips, 2023, p. 463). These properties include among others audiovisual content, installation elements, the overall visual presentation, modes of interaction, and the original source code – with particular attention to the programming language(s) employed, technological choices, and coding conventions (Engel & Phillips, 2023, p. 463).

3.3 Risk Assessment and Crucial Items

3.3.1 Risk Assessment

Risk assessment is a vital step in preserving software-based artworks, as it "[…] supports the development of preservation strategies and helps to prevent damage and loss to objects and collections" (Engel & Phillips, 2023, p. 464). Initially rooted in cultural heritage conservation, risk assessment is a widely used method for "[…] identifying hazards, threats, and risks to objects and collections, evaluating their impact, and finding mitigation strategies" (Engel & Phillips, 2023, p. 464). ICCROM defines risk as "[…] the chance of something happening that will have a negative impact on our objectives" (ICCROM, 2016).

This cultural heritage-based methodology was adapted for contemporary art installations and has proven valuable for software-based art (Falcão, 2010). In digital art preservation, change is considered essential for maintaining functionality, unlike traditional fine art conservation. Another important principle is significance. - Significance, defined as "the values and meanings that items and collections have for people and communities" (Russell & Winkworth, 2009, p. 10), is key to artwork preservation. Artwork elements often possess multiple significances simultaneously. A computer, for instance, can have historical, functional, aesthetic, or conceptual significance, depending on the artwork's identity and work-defining properties. Significance can be attributed by considering software and hardware components. This relative significance often guides recoverability assessments when facing the threat of artwork failure, be it physical or digital. For example, if a computer's significance is primarily aesthetic, replacement with visually similar results is viable. If functional significance is more important, recovery options range from a complete replacement to careful storage. Conceptual significance limits options: preserving the original computer (like a statue) is one possibility. Functionality, which is often crucial for time-based media art, is often prioritised. Artists may accept alterations to maintain functionality, even if it could come at the expense of aesthetic or historical value (Engel & Phillips, 2023, p. 465).

Deterioration, hazards, and associated risks are crucial to consider; other possible concerns range from fire, water, physical forces, pests and electricity, plastics and metals degradation (Waller, 1994 as cited in Engel & Phillips, 2023, p. 466). Digital preservation also addresses software-specific risks like obsolescence, rendering issues, data loss, and inaccessibility (Digital Preservation Coalition, 2022) – these issues will be discussed further in subchapter 12. The degree of risk depends on threat probability, impact, and recoverability likelihood (Engel & Phillips, 2023, p. 466). Obsolescence, defined by Falcão (2011) as "[…] a condition that affects an element or system when it or its parts are no longer commercially available or supported by the industry that initially produced them", is a key risk in regards to software-based artworks. The challenge of obsolescence is not so much about the termination of hardware or software, but rather about the increasing difficulties of replacement or repair and their costs. This incompatibility with modern environments further worsens the risk for obsolescence. The evolution of hardware and operating systems is inevitable, increasing software obsolescence risk. Libraries and drivers become outdated or unsupported, hardware becomes obsolete, and programming languages or operating systems may change

inconsistently, causing software malfunction or failure (Engel & Phillips, 2023, p. 488). The resilience of software applications and programming languages to obsolescence is therefore fundamental for long-term preservation. Problems typically arise when software environments or operating systems update, hindering artwork functionality. Changes to software infrastructure are usually announced in advance, allowing conservation experts time for risk assessment and intervention planning while artworks remain functional. Preservation measures preparing artworks for hardware transformations and for protection against software changes aid in this process. Minimising these risks involves anticipating and proactively preventing software environment changes—disabling automatic updates and carrying out regular backups offer simple solutions (Soares & Simão, 2020, p. 54).

An early assessment during the acquisition of the artworks is crucial to ensure future recoverability. – To achieve this goal, conducting interviews or questionnaires with artists is becoming more common in collecting institutions. These interviews include questions about the work's history, boundaries of the work, important behaviours, essential characteristics, and which specific changes would be acceptable in case of needing to keep the work alive. Other questions are hardware related: they ask about element replaceability, the potential impact on the artwork and if fitting replacements can even be found or produced (Electronic Arts Intermix, n.d.-a).

3.3.2 Crucial Items

A few components are vital for a software-based artwork to have a successful ongoing life within a collection: its software and hardware elements, along with the rights and information secured by collection personnel from galleries or artists.

Acquisition deliverables may vary depending on artwork type, technological dependencies, collection resources, artist preferences, and preservation awareness. Artists should provide comprehensive materials to maintain management and the artwork's integrity, realised through upkeep, examination, and adaptation: Firstly, a complete, operational artwork version, with all hardware and software, which serves as an artist-approved reference. Secondly, technical hardware specifications, including potential replacements. Thirdly, clear installation instructions are needed, covering various conditions, software configurations and operations, and maintenance failure advice, including specialist contact details. Furthermore, a replacement parts stock, if necessary, should also be included. A complete software backup, central for operation and included executables, firmware, and dependencies, is also key.

The project's complete source code, alongside related files and assets must also be provided. Finally, artists should formally grant the institution the rights to publish, exhibit, preserve, and copy the artwork for access, preservation, and necessary treatments (Engel & Phillips, 2023, p. 468). They consider "Obtaining [of] the uncompiled, human-readable source code in addition to the compiled, machine-readable executable file […] [as] […] perhaps the most crucial act of preventive conservation" (Engel & Phillips, 2023, p. 192). Source code analysis has been described as fundamental for preservation, "[…] making it possible to re-create a work, or migrate it to different technologies" (Jarczyk et al., 2017, p. 7). It is vital for maintaining, updating, or migrating software-based artworks; lack of source code significantly increases the risk of losing artwork access. This point will be discussed in more detail in subchapter 3.8.2 ("Source Code Annotation and Preservation") and will also be relevant to the third case study (*Unerasable Character Series*). Beyond source code and executables, further preservation options include creating an identical exhibition copy of the complete system (software and hardware) and, as previously already mentioned, a disk image of either the original computer or a newly created one with the artwork's software environment. Other important considerations include the collection of any miscellaneous, compatible applications or drivers and the documentation of possible registration data. Separating and storing media files away from software minimises the risk of losing access and increases reconstruction possibilities. If the source code is unavailable, decompiling the artist-provided executables may be necessary. Thus, collection experts should obtain decompilers and compilers as precautionary measures. Comprehensive audiovisual documentation, such as narrated screen or video recordings accurately capturing the artwork's experience, behaviours, appearance, and functions, also remain recommended (Engel & Phillips, 2023, p. 469).

3.4 Different Parts of an Artwork and Their Analysis

Software-based artworks generally combine two primary units: software and hardware. Software components consist of encoded instructions and associated data that direct a computer's operations. These instructions are written in programming languages, enabling computer storage and on-demand execution (Soares & Simão, 2020, p. 63). Hardware, in general terms, involves the electronic, physical components of a computer system. In combination with software, hardware components translate instructions and inputs into tangible, visible outputs (Soares & Simão, 2020, p. 62).

Computer capabilities can be extended through peripherals, easily detachable hardware

devices that provide additional input or output functionalities. Common peripherals include digital projectors, audio equipment, cameras, monitors and human interface devices (mice, keyboards, controllers). Recognising and understanding peripherals is vital due to their specific characteristics and potential obsolescence issues affecting long-term functionality (Engel & Phillips, 2023, p. 477).

Connected to the different software and hardware parts making up an artwork, is their analysis and documentation, which will be explained in this next section.

As previously mentioned, “Documentation is the process of gathering and organizing information about a work - including its condition, its contents, and the actions taken to preserve it” (Electronic Arts Intermix, n.d.-b) - this is best done when the system is operational. Capturing precise model details and other key information for all components not only assists backup and disk image creation, defines migration and emulation configurations, but also ensures potential future replacements. Standard off-the-shelf computers often have predefined hardware sets associated with specific models. A thorough documentation still recommends direct examination of the computer with a checklist, encompassing its components, operating system, installed software, and connected peripherals (Engel & Phillips, 2023, p. 478). Software tools, such as *System Information* on macOS and Windows 10, can offer supplementary data. Selecting a tool capable of exporting gathered information in a human-readable format like HTML or XML is advisable.

Visual inspection of internal hardware components may be necessary if software tools are unsuitable.

Checksums and playability of the content are also examined, and it is verified whether the file names match the inventory numbers in the database in order to ensure that the work can be found on The Archive server and that it can be assigned with all associated information (Mürer et al., 2021, p. 41). Collection of any other significant documentation, such as manuals, is also necessary and particular attention should be paid to specific drivers required for proper peripheral function. Generally, the aim is to gather sufficient information for theoretical reconstruction, if necessary (using a template like the *Software-based Artwork Conservation Report* or the *Virtual Reality Artwork Acquisition Template* is recommended at this step) (Campbell & Hellar, 2019).

Soares and Simão summarise it very well:

> The main goal of the documentation created in software-based media conservation is to have a record of the materials that compose an artwork: the artwork's production processes, information about past and present displays and of course, any interventions that may have changed them. (Soares & Simão, 2020, p. 55)

Thorough documentation is essential should hardware elements fail or become obsolete. Hardware component failure can occur from electronic part degradation or inflicted damage. When confronting hardware failure, resolution efficiency is best determined by "[…] balancing the benefit against the resources required" (Engel & Phillips, 2023, p. 481). Soares and Simão report that replacing one layer (hardware) without interference with another layer (software) is often only possible in theory, as the process rarely ever succeeds in practice (Soares & Simão, 2020, p. 51). Despite that, in some instances, simply replacing the failed hardware element with a functional equivalent may still be more convenient and economical than trying to repair it. However, hardware obsolescence complicates this (Electronic Arts Intermix, n.d.-e); parts cease to be sold or supported due to production or support termination, newer parts offering superior performance, or diminished market demand. This not only halts acquisition but also impedes hardware repair due to dwindling expertise and reduced funding.

This often ends in an alternative approach: hardware migration. Migration involves using different hardware replacements, which has a secondary effect on software survival and functionality due to changing hardware support in operating systems: "[…] vendors are unlikely to continue to provide driver software for hardware that is no longer commercially available" (Engel & Phillips, 2023, p. 481).

Software obsolescence and hardware migration have already been explained in subchapter 3.3.1 ("Risk Assessment and Crucial Items") and will be further explored in subchapter 3.7.3 ("Environment Migration").

For now, software in its various facets will be introduced.

3.5 Introduction to Software and Source Code

Software generally comprises code, which can be written in numerous programming languages and is generally described as "[…] a set of instructions, data or programs used to operate computers and execute specific tasks" (Hashemi-Pour, 2024).
Software Heritage defines source code as

> […] a unique form of knowledge which is designed to be understood by a human being, the developer, and at the same time is easily converted into machine executable form. As a digital object, software source code is also subject to a peculiar workflow: it is routinely evolved to cater for new needs and changed contexts. To really understand software source code, access to its entire development history is an essential condition. (Di Cosmo & Zacchiroli, 2018, p. 2)

Artists utilise programming languages based on their specific needs and requirements, instructing computers in their operations. Like all technologies, programming languages have evolved over time, reflecting hardware advancements, computer functions, and their societal roles, fulfilling specific computational and operational demands. Engel and Phillips define source code as

> […] a human-readable set of instructions written in a specific programming language [which] can be read and/or edited in any text editor and is commonly written using a software program called an IDE (Integrated Development Environment), which is designed to support the software development process. (Engel & Phillips, 2023, p. 482)

Source code is of significant value for preserving software-based art, as has already been explained previously.

3.5.1 Different Types of Software

With this basic introduction to software established, we can now examine individual software categories.
A fundamental distinction exists between *off-the-shelf software* and *custom software*. Contemporary artists mostly utilise custom software to realise their characteristic style regarding programming language, algorithm design and coding for artwork aesthetics and performance (Engel & Phillips, 2023, p. 483). Off-the-shelf software is generic, non-individualised, and not explicitly created for a particular software-based artwork. It is typically pre-packaged and broadly accessible, fulfilling universal computing needs.
Another software classification to consider is *open-source* versus *proprietary software*.

Proprietary software is exclusive, only available to licensed customers and copyrighted by a publisher. Even with a license, vendors enforce limitations, generally restricting installations via serial numbers, online authentication, or product keys. Open-source software presents a contrasting model, being freely available and granting source code access. This enables source code acquisition and reuse, which is beneficial for preservation, and permits necessary modifications to further preservation efforts (Engel & Phillips, 2023, p. 484), which will be reviewed further in subchapter 3.6 ("Programming Languages, Platforms and Their Significance Regarding Preservation") subchapter 3.7 ("General Treatment and Preservation Strategies for Software-Based Art"), and in the individual case studies in chapter 5.

3.5.2 Analysis and Documentation of Software

After software type identification, analysis and documentation are necessary. Artwork inspection offers an initial understanding of its composition, but human observation alone cannot fully recognise complex features (such as randomisation or generative functions), making source code analysis necessary.

Software environment documentation requires identifying the operating system and components like drivers and libraries. Crucial for this is classifying the specific operating system and program versions. Built-in and supplementary tools exporting non-proprietary formats (XML, HTML) should be used. Differentiating artist-created from off-the-shelf software within the artwork is also vital since artists often modify default configurations. Documenting custom modifications to off-the-shelf components is important, ideally through collaboration with the creator to understand customisations and their significance.

3.5.3 The Value of Source Code

Software has been a continuously growing feature of humanity's current technological, scientific and cultural knowledge, as Di Cosmo and Zacchiroli have already mentioned in 2018 (Di Cosmo & Zacchiroli, 2018). Therefore, source code examination is not an option but a necessity, revealing algorithms and instructions, uncovering the artists' choices on work behaviour, display, and technological origins.

"This examination method requires the expertise of a programmer or computer scientist and has been introduced to the conservation community in recent years through interdisciplinary efforts between media conservators and computer scientists" (Engel & Wharton, 2014; Engel & Wharton, 2015; Guggenheim Blog, 2016; Engel & Phillips, 2018; Engel & Phillips, 2019, as cited in Engel & Phillips, 2023, p. 484). Examples of

what source code analysis can reveal and enable are investigations of artist-specified colour use, behaviour with interactive devices or animation speed of a displayed object. Beyond these aspects, source code can also identify conservation concerns related to programming languages, technologies, and third-party libraries (e.g., drivers or hardware requirements). It facilitates resource analysis, which is necessary for preservation, and assists the experts in charge in developing an understanding of the artwork's behaviour and risk, enabling informed decisions on possible incidental interventions (Engel & Phillips, 2023, p. 485).

After the source code has been written during project development, a Software Development Kit (SDK) or an Integrated Development Environment (IDE) come into play which include a compiler, to generate an executable file (Lutkevich & Wigmore, 2022). Executable software, therefore, typically contains compiled, human-unreadable code, complicating analysis without this previously mentioned source code. Executables alone pose preservation risks, hindering some strategies. This next section addresses the challenges of examining and preserving executable-only software.

Unlike metadata-rich formats, executable files offer only limited metadata. "Metadata, […] is data about data, documenting technical information about a digital file - everything from the basics such as creation, date, and file format to detailed information on codecs, compression, etc." (Electronic Arts Intermix, n.d.-d). Meaningful information extraction requires debuggers and decompilers, resulting in less comprehensive insights than the source material. - This often leads to reverse engineering, "[…] that is, attempting to gain information about how the software works with limited prior knowledge about how it was created" (Engel & Phillips, 2023, p. 487). Reverse engineering aims to estimate the original source code, a labour-intensive and specialised process. Decompilation, on the other hand, offers a shortcut, using decompilers to approximate source code, with efficacy varying by software and tools used (Castagné, 2013).

3.6 Programming Languages, Platforms and Their Significance Regarding Preservation

Artworks frequently operate using multiple programming languages since one single language may be too narrow for the diverse tasks artists work on. Understanding the languages in use and their specific functions is vital for conservation experts to effectively plan long-term preservation and re-exhibition. Two programming language categories are outlined below, "[…] to point out specific features, functionality, behaviors, and risks associated with each of these languages or language groups in the context of art conservation" (Engel & Phillips, 2023, p. 488).

3.6.1 Interpreted High-Level Programming Languages

The first category is *interpreted high-level languages* or *scripting languages*. These require interpreters for execution. - They perform human-readable source code while the program runs; installing a suitable interpreter (e.g., a Python interpreter for Python scripts) is necessary for this. Scripted languages can run locally or on web servers. "Server-side web-based scripting languages, such as […] Python, are interpreted but the source code is not accessible to the institution or collector unless the artist provides access to the server or the files" (Engel & Phillips, 2023, p. 489). Scripts which are tasked with handling assignments (such as file management) are generally not accessible and must be provided specifically to collecting institutions. Preservation involves collaboration with the IT staff responsible for web server maintenance and stability, as software updates can cause incompatibilities. Python will be investigated further in subchapter 5.4 of Winnie Soon's case study.

3.6.2 Markup Languages

The second category, markup languages like HTML, form a distinct classification. They are not programming languages per se but rather responsible for structuring documents and presentation using tags and attributes to annotate content and differentiate formatting from structure.

HTML (Hypertext Markup Language) defines text, headings, links, and semantics. It is human-readable and often works with CSS (Cascading Style Sheets) for layout, typography, and colour. Text-based HTML and CSS facilitate storage, preservation, and cross-platform compatibility. Their relatively slow evolution, with early notice of changes, facilitates conservator software update management (Engel & Phillips, 2023, p. 489). HTML and CSS will be dealt with in more depth in Chapter 5.4 (*Unerasable Character Series*).

3.7 General Treatment and Preservation Strategies for Software-Based Art

“In the long-term care of software-based art, we must contend with the inevitable failure and obsolescence of components” (Engel & Phillips, 2023, p. 496). - The stable lifespan of software-based art is fundamentally connected to the stability and possible obsolescence of its essential parts. These components will degrade or become outdated over time, requiring a system adaptation to ensure the artwork's continued existence. Important to consider is the work’s fundamental artistic integrity, which should not be compromised. Like many others before them, Soares and Simão (2020) have reported that the modification and management of these components can extend the artwork's life while respecting its original characteristics and design. They find it crucial to acknowledge that the specific technologies utilised in an artwork's creation are fundamental to its historical context and artistic significance. A thorough documentation of these elements is therefore of highest priority, guaranteeing accessibility and informed interpretation if component replacement should become required.

This section will explore various strategic approaches to preserving, treating, and, when unavoidable, intervening in software- and computer-based art. These preservation strategies can be broadly categorised into two primary methods: environment interventions and code interventions.

Environment intervention strategies are characterised by their focus on modifying the hardware or software environment that surrounds the artwork. In opposition, *code intervention* involves directly altering the software code itself to keep up its intended functionality (Engel & Phillips, 2023, p. 496).

It is important to note that the most effective preservation strategy is never the same for different artworks and therefore often entails a combination of both an environment and code intervention technique. The specific approach must be drafted carefully to address the distinctive challenges presented by each individual work of art.

3.7.1 Environment Intervention Strategies

As previously indicated, environment intervention strategies represent a class of preservation approaches focused on modifying or adapting the technical environment within which a software-based artwork runs. A technical environment can be understood as containing two distinct yet connected elements: the hardware environment and the software environment.

The hardware environment contains the tangible physical components of the computer

system, while the software environment consists of off-the-shelf software components distinct from the artist's original software.

These environment intervention strategies can be implemented to address a range of challenges characteristic to the care of software-based art. Most notable are hardware component failure due to incompatibility and the requirement to execute software on systems it was not originally designed for (Soares & Simão, 2020, pp. 51 - 54). The following subsections will explore the various environment intervention methods while taking their limitations into account and outlining universal principles for their application.

Several key considerations should guide the selection process when deciding on the most appropriate environment intervention approach for a specific software-based artwork.

The initial and most fundamental step should always involve thoroughly assessing the possibility of a direct hardware repair or comparable component replacement, as already mentioned in subchapter 3.3 ("Risk Assessment and Crucial Items") and subchapter 3.4 ("Different Parts of an Artwork and Their Analysis"). If hardware repair or replacement is practically feasible, *environment maintenance* is often considered the preferred initial preservation approach. This preference stems from the fact that environment maintenance typically minimises the risk of accidentally introducing unintended changes or modifications to the core artwork itself. However, if direct hardware repair or component replacement is believed to be impractical or unfeasible (due to excessive costs or lack of available parts), conservation professionals should explore the potential of *environment migration*, *environment emulation*, *environment virtualisation* or *containerisation* as a more robust and adaptable preservation strategy. These may all seem very similar at first, but their differences, varying capabilities and specialities will be explained in the course of this chapter.

Methods regarding environment intervention strategies are rarely one-size-fits-all and must be carefully considered on a case-by-case basis, informed by the specific characteristics of each artwork. The conservation concerns cannot be resolved in an abstract manner but need a concrete infrastructure to be addressed appropriately. This decision-making process "[…] will usually involve assessing the dependencies of the software and determining whether or not they can be maintained after the proposed […] process" (Engel & Phillips, 2023, p. 500). In many instances, practical experimentation and testing might be necessary. The reason for this is to correctly predict the potential

impact of any proposed environmental changes on the software's continued working execution and intended artistic expression.

3.7.2 Environment Maintenance

Engel and Phillips (2023) have written substantially about *environment maintenance*. It is often the first preservation strategy being considered since it centres on maintaining the operational integrity of the technical environment. This is to ensure the sustained and successful execution of the artwork's software. Environment maintenance includes routine tasks such as hardware repairs and software updates. While environment maintenance can be highly effective in the short term, its long-term viability faces fundamental challenges. The availability of compatible replacement components, suitable software updates, and the specialist technical expertise often required for maintenance can become progressively limited over time, which diminishes the effectiveness and even the feasibility of this approach. If such limitations become apparent, alternative strategies such as environment migration, environment emulation, or code intervention must be considered as more sustainable preservation solutions.
A classic example of environment maintenance consists of exchanging failed hardware components. A malfunctioning internal computer component can often be replaced with an identical or closely similar device. However, compatibility is vital in such replacements. Factors such as interface compatibility, performance specifications, and driver availability must all be thoroughly evaluated to ensure successful integration and continued functionality.
Off-the-shelf software updates represent another aspect of environment maintenance. This can range from updating device drivers to maintain their compatibility with hardware to applying security patches as protection. However, one thing is central to recognise: software updates, while often essential for security and continued operation, can also introduce unforeseen risks. This is due to the complex dependency relationships between software components and potential alterations to software features (the aforementioned updates), which may impact the artwork's behaviour.

3.7.3 Environment Migration

Engel and Phillips continue with *environment migration*, representing an alternative strategy for mitigating potential artwork loss due to obsolescence.
This approach involves systematically transferring the artwork's software from one technical environment to another, often more modern, environment. Importantly, environment migration aims to achieve this transfer without directly altering the core

software code itself (Electronic Arts Intermix, n.d.-g). This strategy can prove particularly useful in scenarios such as "[…] installing software designed for an obsolete computer system on a contemporary computer system" (Engel & Phillips, 2023, p. 497).

Environment migration eases many of the immediate risks associated with physical hardware component failure and the broader challenges of hardware and software obsolescence. However, the long-term effectiveness of environment migration is often limited by the constantly evolving nature of technical environments. These ongoing evolutions can disturb the dependency relationships between the software intended for preservation and its envisioned technical environment. This fundamental instability often renders environment migration a viable preservation strategy but only for a relatively limited timeframe (Falcão, 2019, pp. 282 - 284).

Judging the practical feasibility of environment migration demands a thorough understanding of the artwork's software dependencies. For example, in the scenario of replacing an ageing computer system, factors such as operating system compatibility or the central processing unit (CPU) architecture are critical to consider. While perfectly matching these elements to the original system would be the ideal outcome, it can often prove very challenging to achieve in practice, primarily due to the increased speed of technological obsolescence. For instance, the Windows XP operating system, which is no longer supported by Microsoft, may lack compatible device drivers for contemporary hardware components. This can render a complete migration of the original software environment impractical or require significant compromises in functionality (Engel & Phillips, 2023, p. 497). If environment migration is deemed not to be a viable preservation option, environment emulation should be explored and evaluated as the next logical step.

3.7.4 Environment Emulation

In contrast to migrating software to a fundamentally new technical environment, emulating an appropriate technical environment often poses a more strategic preservation approach.

Environment emulation involves employing specialised hardware or software tools known as *emulators*. These tools accurately simulate the operational properties of one computer system on a fundamentally different system. This should enable the execution of legacy operating systems and software on modern hardware platforms for which they were never originally designed (this will be thematised in more detail in Chapter 5.1).

Environment emulation offers significant benefits for long-term digital preservation. For the most part, it allows continued access to platforms that have become technologically obsolete. This is achieved without requiring the gradually more complex and often impractical acquisition and repair of the original physical hardware. Emulation, as a broad preservation strategy, encompasses several related but distinct technical approaches. In its strictest sense, a *full-system emulation* entails fully simulating all hardware components and functionalities within the emulated environment (Electronic Arts Intermix, n.d.-f; Falcão et al., 2016).

Software tools designed to run on modern host computer systems currently represent the most practical and widely adopted option for achieving desktop computer emulation. Within the context of software emulation, specific terminology is used. The physical computer system on which the emulation software executes is conventionally termed the *host* system. The computer system being simulated by the emulator is referred to as the *guest* system (Engel & Phillips, 2023, p. 498).

Below is an exemplary case of how an emulation workflow could play out.

Example of an Emulation Workflow:

A diverse range of tools and established workflows exist for implementing environment emulation strategies. This subsection presents a simplified, step-by-step workflow approach based on the established emulation workflow employed at Tate Modern, one of the world's largest museums of modern and contemporary art. This workflow was rigorously developed and refined during the Software-based Art Preservation Project (Ensom et al., 2022) and is further informed by extensive research conducted by Klaus Rechert and Tate in 2016 and Engel and Phillips (Engel & Phillips, 2023, pp. 501 - 503; Rechert et al., 2016).

1. **Emulator Selection:** This initial and crucial first step involves carefully selecting an appropriate emulator. Identifying the specific processor architecture and operating system for which the artwork’s software was initially designed provides a logical starting point for emulator selection. Suggested open-source emulator options of classic Macintosh computers, often well-suited for art preservation contexts, include *Basilisk II*, *PCem*, *QEMU*, *SheepShaver* and *VirtualBox* (E-Maculation, 2025) or *Stella*, *STeem* or *SainT* for Atari machines (Mee, 2024). When selecting an emulator, considering options offering hardware virtualisation support for a potentially better emulation performance can be a worthwhile consideration.
2. **Disk Image Setup:** “A disk image […] will act as a storage device for the emulated computer system” (Engel & Phillips, 2023, p. 501). After a disk image of the

artwork’s original hardware system has been successfully acquired, it is recommended to assess whether the acquired image needs any format conversion. This is to crosscheck the compatibility with the previously chosen emulator. If converting the format is not required or has already been performed, utilising the original disk image is generally recommended.

If an original disk image is unavailable, the suggested alternative approach is to create a synthetic disk image specifically for use with the chosen emulator. This is typically achieved by installing an operating system and any other necessary supporting software components directly within the emulated environment.

3. **Emulator Configuration:** Once a suitable disk image is prepared and integrated within the selected emulator, it is recommended to adjust the emulator's configuration settings. For this, it is essential to mirror the original specifications as closely as possible to ensure optimal emulation accuracy and software performance.
4. **Emulation Execution:** With the emulator configured and the disk image prepared, the next step involves initiating the emulation process and attempting to boot the artwork’s software within the emulated environment. If the software fails to boot successfully during this stage, one should systematically revisit the emulation workflow's preceding steps. This iterative process may involve carefully re-examining and adjusting the emulator's configuration settings or potentially modifying the disk image itself to address any issues.
5. **Emulation Assessment:** Once the artwork’s software is successfully running within the emulated environment, the next step is to thoroughly assess the emulation's accuracy. This assessment phase involves meticulously comparing the experience of interacting with the emulated artwork to the intended or documented behaviour of the artwork running on its original hardware. This comparison can be made using various methods: direct side-by-side comparisons with the original physical hardware (if available), using existing video and image reference materials that document the artwork’s original appearance and behaviour, or using quantitative metrics where possible to evaluate performance characteristics objectively. The use of an emulation report is thoroughly recommended at this step; an example of a report would be the template devised by Tate Modern (Time-based Media Conservation, Tate, 2020b).
6. **Emulation Documentation:** Comprehensive documentation of the entire emulation process is essential for future reference and ongoing preservation management. This documentation should record key details, including the specific workstation

hardware used for the emulation, the precise versions of the emulator software and guest operating system employed, any modifications made to the disk image during the setup process, detailed emulator configuration settings, and comprehensive notes documenting any challenges encountered during the emulation process, as well as any other notable observations (Time-based Media Conservation, Tate, 2020b).

7. **Emulation Archiving:** To facilitate potential future repetition of the emulation process and to ensure long-term preservation and accessibility of the emulated artwork, all essential components of the emulation setup should be archived systematically. This archival process should include the disk image file, the specific version of the emulator software used, and all associated documentation generated during the emulation workflow. Systematic archiving of these components is particularly critical for enabling accurate future display of the artwork and for supporting any necessary preservation interventions that may be required in the future.

Even unsuccessful emulation attempts can produce valuable knowledge and insights about the software-based artwork, which has been the case for *TwiLight*. By systematically testing different emulation approaches and configurations, conservators and preservation professionals can gain a deeper understanding of the complex relationship between the software and its intended technical environment, even when a complete emulation remains unsuccessful.

3.7.5 Environment Virtualisation

While *environment virtualisation* resembles emulation to a certain extent, there exists a significant difference between them: virtualisation offers the guest operating system direct access to the underlying physical hardware of the host system to a much greater extent than emulation ever could. This difference is particularly pronounced in *Central Processing Unit* (CPU) *virtualisation*, a common virtualisation technique.
In CPU virtualisation, a specialised software component – a so-called *hypervisor* – enables virtualisation software to execute code directly on the host machine's physical processor (vmware, n.d.). An important condition for effective virtualisation is that the guest operating system must employ the same *Instruction Set Architecture* (ISA) as the host processor. Unlike emulation, where processor instructions are typically translated or interpreted by the emulator, virtualisation avoids direct processor instruction translation. This difference results in significant performance advantages for the guest system running within a virtualised environment, which allows software to execute at

speeds much closer to the original hardware performance (Engel & Phillips, 2023, p. 499).

Virtualisation is frequently used interchangeably with *CPU virtualisation*. However, its functionality can be extended further to other hardware components, such as the *Graphics Processing Unit* (GPU). This is commonly known as *para-virtualisation*, a technique that allows certain hardware components to be efficiently shared between the host system and one or more guest systems.

Passthrough is another related virtualisation technique. It further refines hardware access by providing guest systems with direct, exclusive access to specific host hardware components, enabling them to operate at native hardware speeds. However, this direct passthrough approach limits the ability to share the designated hardware component with the host system or other guest systems. More significantly, from a preservation perspective, passthrough can introduce stricter hardware and operating system dependencies within the guest system, which may have implications for long-term preservation prospects (Sheldon, 2024; Engel & Phillips, 2023; Friel, 2020).

3.7.6 Containerisation

Containerisation, or *operating system-level virtualisation*, shares some conceptual similarities with traditional virtualisation techniques. Containerisation involves packaging software and all its essential dependencies into a self-contained unit, a *container*. These containers can be readily shared and reused across different computing environments. This packaged form is commonly known as a *container image*, and these images can comply with various standardised file format specifications to enhance portability and interoperability. Similar to virtual machines, containers have found widespread applications in web development and implementation contexts. – They offer the practical ability to operate multiple isolated software environments simultaneously on the same physical server infrastructure. However, in contrast to virtual machines, container images do not include an operating system within the container itself. This architectural difference results in container images being significantly smaller in storage footprint compared to virtual machines. Containers rely on specialised container *runtimes*, such as *Docker Engine*, instead of a complete hypervisor to manage and execute the contained software applications (Hashemi-Pour et al., 2023; Mell, 2023; Sheldon et al., 2024).

Containerisation has emerged as a valuable technique for web-based artwork presentation and preservation. For example, *Rhizome*, a digital art organisation

referenced numerous times so far, is using containerisation to provide ongoing access to older versions of web browsers, which they need to run legacy web-based artworks properly (Rossenova, 2020). Similarly, the *Haus der elektronischen Künste* (HeK) in Basel uses containerisation strategies for efficient and stable management of their web server infrastructure. In this context, each individual artwork within the HeK's collection is typically deployed and executed within its own dedicated container. This container-based isolation ensures compatibility and prevents potential software conflicts between different artworks, even when they rely on conflicting or potentially incompatible software versions. However, it is critical to note that containerisation "[…] does not have the preservation applications of emulation or virtualization, […], which achieves a level of independence from the physical hardware" (Röck, 2020 in Engel & Phillips, 2023).

In conclusion, one can see that environment maintenance focuses on preserving the original operational integrity of the technical environment as much as possible, while environment migration systematically allocates the artwork's original software from one technical environment to another. Emulation and virtualisation aim to abstract the software environment from the underlying physical hardware. Containerisation, however, inherently relies on a functional container engine. It also requires that the software contained within the container image remain compatible with the host operating system kernel of the system on which the container is eventually executed.

3.8 Code Intervention and Preservation Methods

After discussing environment intervention techniques at length and identifying their prospective strengths and weaknesses, the focus will now lie on code intervention and preservation methods in general. Possible intervention strategies regarding specific source code files will be found in greater depth in Chapter 5 (specifically by investigating the artwork series by Winnie Soon).

Code intervention, the direct modification of the software code itself to maintain its accessibility and functionality, may become a necessary approach to successfully restore and preserve software-based art. This might be required in certain instances, particularly when environment intervention strategies, as discussed before, prove insufficient or impractical. However, carefully assessing the extent of the required code intervention is of utmost importance. In some cases, a relatively minor adjustment may be enough to restore functionality. One instance might be to modify a single line or a few lines of scripting language code to update a URL linking to an external dependency

(Engel & Phillips, 2018).

More complicated code modifications mean a significantly more resource-intensive preservation endeavour. An example of this might be a complete migration of software code from an obsolete programming language to a contemporary language and development environment. Such vast code interventions should only be taken on after a careful and thorough evaluation of the potential benefits of code migration compared to the associated risks and demands (Engel & Phillips, 2018).

3.8.1 Code Migration

Code migration is defined as the systematic process of transferring software functionality and logic from one programming language or development environment to a fundamentally different one (GeeksforGeeks, 2024). Although it remains a valuable method in the preservation toolkit for software-based art, it is important to recognise that code migration can be a resource-intensive endeavour. Numerous factors come into play: one is prioritising minimal intervention in the original code another is respecting the artist's original artistic choices and intentions embedded within the software. Additionally, ensuring the reversibility of any code modifications made and carefully selecting a target programming language and development environment that evidently supports long-term software preservation and maintainability (Engel & Phillips, 2023; Soares & Simão, 2020, p. 50) are also essential to consider.

Various practical considerations should inform the selection process when selecting a target programming language for code migration. Firstly, it is relevant to consider whether a newer, still-supported version of the original programming language or development environment might be available. - If a newer compatible version exists, upgrading to this variety may represent the best approach, as it preserves the original technological choices to the greatest extent possible. However, if a compatible updated version of the original language is not available, conservation personnel should explore alternative, but functionally similar, programming languages with comparable syntax, semantic structure, and development patterns to the original language.

Software compatibility must be carefully assessed in the context of code migration. This assessment ensures that all necessary software libraries and external dependencies required by the original software are available and compatible within the new target programming language and development environment. Furthermore, a detailed evaluation of how the proposed code migration could potentially impact the artwork's interaction with both its hardware and software environments is critical. This evaluation

should identify whether the code migration process will necessitate any adjustments to the artwork's environment (Engel & Phillips, 2023, p. 504).

3.8.2 Source Code Annotation and Preservation

Besides code migration, source code preservation also exists as a code intervention method. This chapter is split into two parts: the first one being an introduction to source code in general and what is currently understood as source code annotation. The second part deals with a workflow of what source code preservation could look like.

a. Source Code Annotation

Chapter 3.5 has already breached the topic of Source Code, but the topic will be discussed in greater detail on the following pages. I will start by defining source code, to later expand on source code preservation in particular.

GNU defines a work's software source code as "[…] the preferred form of the work for making modifications to it" (GNU, 1991). Software Heritage expands on this definition: […] software source code is a unique form of knowledge which is designed to be understood by a human being, the developer, and at the same time is easily converted into machine executable form. As a digital object, software source code is also subject to a peculiar workflow: it is routinely evolved to cater for new needs and changed contexts. To really understand software source code, access to its entire development history is an essential condition (Di Cosmo & Zacchiroli, 2018).

As illustrated by the definitions above, source code fulfils three functions:

1) The representation of knowledge must be readable by humans, developers, and machines simultaneously.

2) It must constitute the preferred form of a work, which must still be changeable and consequently needs to be able to be further developed and adapted again and again,

3) while also providing as limitless access as possible to its previous development history.

Source code is generally written in specific programming languages and typically stored in a text file to be human readable. Comparable to any other language, they can be read by those who understand source code – with one of the main differences to spoken languages being, that programming languages emerge and fade much faster than humanity's regular languages, which causes them to be unintelligible for new programmers who were not trained in time to read and understand them. This is why technical documentation of source code provides immeasurable information for future programmers to comprehend how artist-generated artworks function (Engel & Wharton,

2015, p. 92). The documentation will be essential to maintain the software should changes to its underlying operating system be required; works including multimedia items often need technical documentation if changes to relevant standards, image, video or audio formats need to be implemented.

“Software maintenance is a standard practice in software engineering that seeks to correct and improve upon computer programs over the time that they are in use, and technical documentation of a program’s source code is a core activity of this discipline” (Ali, 2005; Das et al., 2007; de Souza et al., 2005; Scanniello et al., 2010; Stroulia & Systä, 2002; Wong & Tilley, 2002 in: Engel & Wharton, 2015, p. 92). Overall, there exist various methods of documentation: code annotation, narrative description, visual documentation and UML diagrams. This next section will focus on code annotation so as not to go beyond the scope.

Figure 1 depicts an example of documentation in which comments were added to the source code itself.


```python
originalsrc = 'generateddata/cleaned_COVID_censored.txt'
filepath1 = 'generateddata/cleaned_1.txt'
filepath2 = 'generateddata/cleaned_2.txt' #final result for use

df = pd.read_csv(originalsrc, engine='python', usecols=['created_at','text_cleaned'])
df.sort_values(by=['created_at'], inplace=True) #sort the date
df.created_at= "</del><timestamp>" + df['created_at']
df.created_at=df.created_at.str.slice_replace(36, repl=' </timestamp><del>') #25

# checking on specific context: this dataset and the characters that are found in this dataset

white_list=['!','(',')','-','[',']','{','}','，','；','：','\'','"','\\','.','?','@','#','$','%','^','&','*','_','~','，','。','：','【','】','"','"','

# special handle '<'
df.text_cleaned=df.text_cleaned.str.replace(r'<(?!/)', '</del><<del>', regex=True) # find symbol < with the next character not equal to /
# special handle '>'
df.text_cleaned=df.text_cleaned.str.replace(r'(?!l)>', '</del>><del>', regex=True) # find symbol > with the previous character not equal to 'l'
# special handle '/'
df.text_cleaned=df.text_cleaned.str.replace(r'/(?!del)', '</del>/<del>', regex=True) # find symbol / with the next character not equal to /

for x in white_list:
    df.text_cleaned = df.text_cleaned.str.replace(x, "</del>" + x + "<del>") #check characters

df = df['created_at'].map(str) + df['text_cleaned'].map(str)
#df = df['text_cleaned'].map(str)
df.to_csv(filepath1, encoding='utf-8', index=False) #without the index column
print(df.head(100))

# combine all rows
data=open(filepath1).readlines()
for n,line in enumerate(data):
    if line.startswith("line"):
        data[n] = "\n"+line.rstrip()
    else:
        data[n]=line.rstrip()
arrayStr = ''.join(data)
#clean tags with duplication
arrayStr=arrayStr.replace('<del></del>', '') #delete tags found in duplicate whitelist (there is a special character in between)

with open(filepath2, 'w+b') as f:
    f.write(arrayStr.encode())
```

Figure 1: Python source code written by the artist Winnie Soon to process the COVID-19 data to be later displayed in her artwork Unerasable Characters III.

Annotations like these can be pasted into blocks of code, specific statements or other structural sections within the code. The source code in black, blue and red font depicted in the screenshot above will be compiled into software, transmitting information to the computer to produce the resulting artwork ending up visible on the computer screen. The text in grey font, preceded by the hashtag (#) marks the technical documentation, assisting future preservation staff and programmers in interpreting the code. When the program runs, the computer knows to ignore the comments marked with # embedded in the artwork's source code.

Code annotation offers a high level of detail (as shown above by the line-by-line documentation), providing future programmers with the information necessary to update, re-compile or migrate the artwork for future display (Engel & Wharton, 2015, p. 54)

Now, how can source code best be preserved for future use?

Because of its human-readable documentary structure, it is an often overlooked aspect of digital preservation strategies. This preservation aspect tends to be difficult abroad the entirety of the digital field, but it becomes especially challenging in cases where the software that was used to create an object or an artwork is no longer current (as it was the case for a chosen artwork, *TwiLight*, which has been mentioned various times before and will be explained in greater detail in a later chapter). The consensus in this case is that a proactive approach to software preservation is of utmost necessity and that "passive gathering of software is not likely to produce a comprehensive and relevant collection, nor can it ensure that the software will perform accurately when needed" (Zabolitzky, 2002, p.8 in: Castagné, 2013, p. 8).

b. Source Code Preservation

How can this proactive approach to preserve software be taken, one might ask?

It all starts with clean documentation – the preparation of a README file with the most important information about one's repository (this includes code and file contents) should be taken as a first step. File formats such as .txt or .md are readable by a broad variety of operating systems, granting them greater longevity and a higher likelihood of successful preservation. The file should contain the most necessary information: the names of the author(s) and their contact details, the title of the artwork and a brief description of what it is supposed to portray. Noting who was responsible for writing the code (as this might be a different person than the artist) and which code version was used are also important metadata. An overview of the files and folders should not be

missing, in combination with workflow instructions on how to run the software and a possible file informing about the requirements, used packages and software versions coupled with their dependencies (Bolnick et al., 2023).

Having organised a proper file layout and documentation, the next step is to revise the code. Any code should be thoroughly annotated with in-script comments describing the purpose of the commands. Scripts should automatically begin by loading necessary packages and by "[…] importing raw data in a format exactly as it is archived in the [your] data repository" (Bolnick et al., 2023). As with every software, artwork code should be tested one last time before upload into a repository – this best happen on a completely empty machine, with no required packages installed.

After having prepared the files, the code and data (if any data was used) are ready to be archived. – The best option for this is a version-controlled repository, like DRYAD[2] or Zenodo[3] (Bolnick et al., 2023), since they place great emphasis on long-term preservation and archiving. GitHub can also be an option, but only under specific circumstances: the GitHub repository needs to be linked to a third-party repository, such as Zenodo.

This is because GitHub does not qualify as a public archive on its own since it does not generate a DOI. Access rights still lie with the person who uploaded the data in the first place, which might cause it to be taken down or otherwise restricted at a later point (UCL, 2018).

Providing as much requested information and metadata as possible makes The Archived files simpler to locate and comprehend (Bolnick et al., 2023). Furthermore,

"[…] access to source code is a major factor in a preservationist's ability to recreate adequate software performance and, to this end, open standards must be actively promoted, regardless of which preservation approach currently seems best. Additional requirements include a strong digital preservation framework that is tailored to the growing complexity of software […]" (Castagné, 2013, p. 8).

The best way to ensure that files will retain their value and use over time is to "[…] prepare and deposit the file formats with the highest probability of long-term preservation" (Mortimore, 2015). When the file formats possess the following characteristics, their likelihood for a successful long-term preservation of content is much higher (this is especially important in circumstances like the case study of *VR Tour through the Doomsday Clock* in Chapter 5.3, where only documentation files exist

[2] https://datadryad.org/stash
[3] https://zenodo.org/

and none of the actual project files are accessible). Since non-text format files (video files, spreadsheets and databases, images, audio and software) are especially vulnerable, additional precautions need to be taken in their regard for a successful and long-lasting preservation. Mortimore and Johns list seven criteria for submitted formats which heighten the likelihood of long-term preservation of functionality and content (Johns, 2025; Mortimore, 2015):

- complete and open documentation
- platform-independence
- non-proprietary (vendor-independent)
- no "lossy" or proprietary compression
- no embedded files, programs or scripts
- no full or partial encryption
- no password protection

File formats can generally be organised by their probability of long-term preservation of content and functionality. Three different levels of probability exist: high, medium and low probability. Mortimore and Johns recommend that scientists or research personnel uploading their works to a repository do so in the following recommended formats and to convert file formats with a lower probability of long-term preservation to a format with a higher likelihood (Johns, 2025; Mortimore, 2015). Based on these criteria, I have decided to categorise each file from each case study's artwork. This categorisation will be demonstrated in the fifth chapter, "Case Studies".

3.9 Concluding Remarks

Ultimately, ensuring to preserve "[…] the experience of an artwork and at the same time its technical history" (Falcão, 2019, p. 278) are the two main objectives of any environment or code intervention strategy. The migrated or emulated artwork must clearly maintain its intended visual appearance, interactive behaviour, and overall artistic impact. Given the central role of long-term preservation in conservation practice, the sensible choice of a new programming language or environment approach is of significant importance. Selecting technologies beneficial to long-term maintainability and forward compatibility helps minimise the potential need for future preservation interventions.

To ensure a successful code migration and to maintain artistic integrity, strict quality control procedures and complete documentation practices are essential throughout the preservation process. Standard software development testing practices should be

implemented throughout the code restoration process to achieve the highest possible quality in the restored artwork. This mimics quality control protocols commonly applied in other art conservation treatments. Systematically testing compatibility across a range of commonly used web browsers and OS platforms (macOS, Linux, or Windows) is especially important for containerised artworks, such as those preserved by Rhizome or the HeK. Additionally, if needed, methodical comparisons between the original artwork and the newly restored or migrated version can be incorporated efficiently into the overall quality control process (Engel & Phillips, 2023, p. 505).

Finally, all code interventions performed during the software migration process should be documented directly within the source code of the restored artwork copy by systematically annotating the code in a clear and informative manner (Engel & Phillips, 2023, p. 505).

4 The Ars Electronica

This section serves as a digression about the institution of the Ars Electronica and my specific motivation and connection to them. The chapter will start by explaining the basic structure of Ars Electronica in general, its history and the museum; connected to this is also The Archive and the database in active use. Further elaboration will be about the Futurelab, the Ars Electronica Festival and the Prix Ars Electronica.

My personal motivation for Ars Electronica stems from my involvement in The Archive department and the 2024 concluded "Relaunch Ars Electronica Archive" project.

The project took on one of the central tasks of digital media art: the professional documentation and long-term preservation of digital cultural heritage. At a time when digital media are increasingly shaping the art and cultural landscape, the cultural sector is faced with the challenge of archiving, preserving and making digitally born content accessible. In particular, the complexity and diversity of physical and digital formats require innovative approaches and solutions for the preservation and cataloguing of these holdings.

One of the main goals was to implement a new workflow for archiving and digitising materials (analogue as well as digital). This included the processing of one of Ars Electronica's founding fathers' estate, which consisted of photographs, videos and documents, as well as the archiving of digital content from the Prix database.

The new digitisation workflow for documents was broken up into four different steps: the starting point consisted of screening and metadata collection, after which the filing and preparation of the documents into archive boxes was taken on; a representative selection of 20% from each document box was digitised and saved (with backup and restricted access). The third step was to record the metadata and upload the digitised versions to the database, assigned to the corresponding boxes. The last step considered copyright and data protection – only cleared documents were made publicly available. Digitising was done by scanning the documents at 300dpi in PDF-A format with OCR enabled to ensure their searchability. The selection criteria for the digitisation itself were a combination of historical value and the need for long-term preservation.

The workflow for digitising photographs followed a very similar sequence as the documents. The photos were viewed, measured and counted to get an accurate impression of the collection, while simultaneously compiling initial metadata about the people and events depicted. The photographs were then freed of any remaining materials and placed in acid-free sleeves. Each photo box was labelled after viewing and

analysis. As with the documents, the metadata collected for the photo boxes was entered into the database, followed by a final rights check regarding publication. The first screening resulted in 686 photographs, of which only a significantly reduced number were digitised due to many duplicates.

The workflow for digital-born material was structured differently. In 2023, the Prix database was slightly redesigned to enable more precise metadata for future submissions and to reduce manual effort. Prix Ars Electronica material was firstly imported from the Prix database, with the submission metadata already entered by the artists themselves and supplemented by the Prix team with additional information. After a comprehensive check and possible corrections by the Prix team, the data was transferred to The Archive database and ultimately checked there. The winners' projects from 2020 to 2024 were used, which amounted to a total of 8789 digital copies of digital-born material recorded and provided with metadata.

The archiving and preservation of software-based artworks were not part of this project's undertaking, which leaves Ars Electronica even more room to grow and develop new techniques and methods for the conservation of digital cultural heritage.

4.1 About the Ars Electronica

4.1.1 General

Ars Electronica is an institute with a focus on education, culture and science, active in the fields of new media art and new technologies. Founded in Linz, Austria, in 1979, it is nowadays based at the Ars Electronica Center (AEC) by the Danube. The institute's activities focus on the collaborative aspects and ties between technology, art and society. This makes it a famous institution in the art world, renowned for organising the yearly known worldwide festival, the Ars Electronica Festival where the Prix Ars Electronica is awarded. Ars Electronica's archive is additionally appreciated as one of the world's largest archives on digital media art, with a user base ranging from school classes and interested people to artists and professionals active in the field of new media art. They also manage the multidisciplinary facility, focused on research and development in the field of new media arts, known as the Futurelab.

4.1.2 History

Ars Electronica was brought to life with its first festival in September 1979, founded by electronic musician Hubert Bognermayr, cyberneticist/physicist Herbert W. Franke, then director of the ORF—Austrian Broadcasting Company's Upper Austria regional

studio - Hannes Leopoldseder and music producer Ulrich Rützel (Ars Electronica Linz GmbH & Co KG, 2022c). Until 1986, the festival was held every second year and has been held every year since 1986. The Prix Ars Electronica was firstly introduced in 1987 and has since then been awarded annually.
The AEC, the museum itself, and the Futurelab firstly opened in 1996, and again after a significant, two-year renovation in 2009, taking approximately two years to complete and costing roughly 30 million euros. The current museum footage holds 3000 square meters for exhibitions, 400 m^2 for conferences and workshops and about 100 m^2 for development and research (Ars Electronica Linz GmbH & Co KG, 2010b, Hirsch, 2019, p. 140).

4.2 The Museum and the Futurelab

The current AEC comprises four building parts: the original building, a new "twin-tower", the main gallery and a bigger, more modern space for the Futurelab, where The Archive is also located. 1.100 glass panels, making up 5.000m^2, cover the whole outside of the building, giving it the appearance of glass skin. – Every single panel has a built-in LED bar, allowing them to change colour individually; this makes the whole building function as a huge display. Artists can create and submit their own graphics and visualisations, which are regularly shown, in particular during the festival.
The Futurelab's goal, as mentioned before, is to "[…] create works that reveal the transformative force that emerges when art, technology, and society converge" by using these works "[…] as a catalyst for future innovation and societal change" (Ars Electronica Linz GmbH & Co KG, 2023d). This is, among other things, achieved by the artistic research and development laboratory character of the Futurelab - their goal is to inspire the public and humanise technologies in collaboration with global partners. As a result, they produce concrete future prototypes and experiences, including immersive artworks, workshops, experiments, technical solution approaches, and interactive installations. The Ars Electronica Futurelab is a mirror for the various futures that lie ahead of humanity, acting as a catalyst and compass (Ars Electronica Linz GmbH & Co KG, 2023d).
Through the combination of practical problem-solving and artistic research, the Ars Electronica Futurelab challenges conventional thinking, regularly coming up with creative ways to start a conversation with society about the many possibilities the future holds. The interdisciplinary team works with various technologies, including robots, swarm intelligence, creative artificial intelligence and microbes, data visualisation,

media architecture, mixed reality, and more. Being a part of a vibrant artistic community that is particularly accessible and of considerable public interest is one of Ars Electronica's hallmarks. - The three foundations of the Futurelab's work are art, technology, and society, which is also the slogan of Ars Electronica (Ars Electronica Linz GmbH & Co KG, 2023d).

4.3 The Archive

The Archive of Ars Electronica belongs to the world's largest on digital media art spanning from 1979 to today. It includes documentation on the Ars Electronica Festival, the artists' submissions in the Prix Ars Electronica archive as well as "[…] documentation on activities of the Ars Electronica Futurelab, the Ars Electronica Center and all other areas of Ars Electronica" (Ars Electronica Linz GmbH & Co KG, 2024b). A big part of the immense archive collection is accessible to the public, thanks to several digitisation projects.

"Ars Electronica's archive contains a diverse assemblage of artistic works and documentation of projects, exhibitions and activities over a timeframe of more than four decades in the wide-ranging international field of media art" (Ars Electronica Linz GmbH & Co KG, 2024a).

Through the yearly festival and other continuously ongoing events, The Archive keeps growing. - It therefore not only offers a "[…] representative cross-section of the broad field of media & digital art" (Ars Electronica Linz GmbH & Co KG, 2024a), but it is also a "[…] historical catalog [sic] of the data storage media and formats that in many cases significantly determine the appearance of the works preserved on them" (Ars Electronica Linz GmbH & Co KG, 2024a). Everything remotely imaginable connected to new media art is stored in The Archive: scholarly articles on computer animation, computer and digital music, computer graphics, net based or interactive art, virtual reality applications, software, media performance, and bio art and robotics "[…] explicitly document the aesthetic strategies used in these fields as well as, implicitly, the technical conditions on which they are based" (Ars Electronica Linz GmbH & Co KG, 2024a). However, given the short half-life of the data storage devices it contains, the Ars Electronica Archive is not only important from a scholarly and historical standpoint, but it also represents an urgent responsibility. The video library comes in a variety of formats, some of which are becoming less and less accessible due to the continuous development of substitute technologies. The same holds true for the

programming languages used to create art projects and the different kinds of storage media they use, such as CDs, floppy disks, and websites, along with their URLs. Besides thousands of audio and video recordings, countless photographs, slides and negatives are also stored in The Archive, next to a broad collection of printed material, including posters, press material, folders and publications. Due to the renovation and the reopening of the AEC in 2009, The Archive was for the first time climate-controlled – benefitting the approximately 60.000 physical records, located in more than 378 metres of shelf space (Ars Electronica Linz GmbH & Co KG, 2024a).

Video documentaries since the festival's inception in 1979 make up a sizeable part of the collection. Thanks to the efforts of the ORF, Austria's national public broadcaster, much of the early programme was carefully recorded and archived. This laid the groundwork for the massive amount of media materials that were to follow. Furthermore, video footage of the festival and the museum's exhibitions has been shot numerous times, adding to the around thousand hours of digitised video material. The enormous diversity of the content's physical and digital formats matches its wide range of subject matter, posing a significant challenge for archivists. While tangible artifacts can be preserved using well tested techniques, digital data preservation for an extended period still presents several challenges (Ars Electronica Linz GmbH & Co KG, 2024a).

The most extensive digitisation and archiving project of Ars Electronica since the 1990s was started during the building of the new AEC in 2006 with the goal of centrally and specifically inventorying and digitising content. The Ars Electronica Archive has therefore undergone physical and digital development and reconfiguration.

“A server-based digital archive has been developed which is based on a complex database structure and metadata associated with digital objects (video, audio, image, text)” (Ars Electronica Linz GmbH & Co KG, 2024a). With a storage capacity of 62 terabytes, it now has over 173.900 entries. Every shared and produced file, together with the database entry, is immediately moved to a backup system and tested. The archival holdings are guaranteed to be protected in the long term by an extra tape backup. Continued action in this domain is essential to achieve a potential plan for the extended preservation of digital information.

“The purpose of the Database is inventory and scholarly research, with the objective of reprocessing and agency´s [sic] the history of Ars Electronica and digital / electronic Arts in general” (Ars Electronica Linz GmbH & Co KG, 2024a). A portion of The Archive is made available online, where artists and scientists are encouraged to interact with the material, provided that doing so is both legally and technically feasible.

4.3.1 Ecosystem of the Database

The Archive system developed represents a comprehensive database solution that stores various multimedia data formats such as text documents, images, videos and other digital formats. This data is stored as archive material along with the associated metadata in the database and can be managed and supplemented in it. Access to The Archive is browser-based, enabling both recording and research via standard web browsers. Depending on the authorisation level in the network, users can add new data, edit data and query existing content.
The system distinguishes between a public and an internal area:

- Internal area: This is intended for data collection and maintenance and enables unrestricted access to all archive data.
- Public area: Selected data from the internal area can be marked as "public" and is openly available in this way. This data is accessible via the online archive, in addition to possibly being displayed on the Kulturpool platform after being marked as such by selected archive personnel.

The system software features an application framework recently upgraded to the current long-term support version of Django 4.2 and a newer database version of PostgreSQL. This not only ensures that future requirements and extensions can be implemented efficiently but also significantly improves the performance of the entire platform.
A newly drafted data model was developed to optimise performance and expandability and meet future requirements. A formerly particularly complex and resource-intensive tree structure previously used to locate and allocate the archival materials has been simplified. A defined tag list was implemented to improve user-friendliness and ensure precise content-related allocation of the archival materials, enabling simple and efficient searching and filtering (Ars Electronica Linz GmbH & Co KG, 2023a).
In addition, the option of creating "project containers" was implemented to facilitate linking and merging objects in the newly designed database. Thanks to their open and flexible structure, these containers can be used for a wide range of applications.
This new concept makes it much easier to manage the various types of archival materials—from physical materials to digital copies and digital-born content—efficiently and appropriately and to relate them to one another if necessary (Ars Electronica Linz GmbH & Co KG, 2023a).

4.4 Ars Electronica Festival

The Ars Electronica Festival, featuring one of the artworks discussed in my case studies, has been happening every first week of September since 1979 (Ars Electronica Linz GmbH & Co KG, 2022d). More than 1.000 artists, designers, entrepreneurs, scientists, and activists from over 40 countries meet in Linz to address and discuss central questions regarding humanity's future and to present their inspiring artworks, unusual prototypes, and groundbreaking research (Mission, n.d.). The festival is dedicated to a yearly new theme, with an exciting array of unconventional, but public, festival locations – a monastery, a tobacco processing plant, Linz Harbor, a network of subterranean tunnels and the former, now unused except for the Festival, Austrian Postal Service logistics centre (Ars Electronica Linz GmbH & Co KG, 2022c).

4.4.1 Prix Ars Electronica

The Prix Ars Electronica, given out annually in different categories, was established to honour innovativeness and creativity in the use of digital media - "The Prix Ars Electronica is the world's longest-running media art competition. With the award-winning works of international artists as a trend barometer, it offers an inspiring, current and forward-looking insight into the interface between art, technology and society" (Ars Electronica Linz GmbH & Co KG, 2022b). Not only that, but the Prix Ars Electronica can be considered "the most traditional media art competition in the world" (Ars Electronica Linz GmbH & Co KG, 2022b), being one of the longest running and best known yearly awards in the field of interactive and electronic art, computer animation, music and digital culture (Ars Electronica Linz GmbH & Co KG, 2010a). Grover described the Golden Nica (the highest award there is to receive), in regard to creativity and pioneering spirit in the digital media, as "the Oscar of digital art" (Grover, 2008, p. 55).

With the first prize awarded in 1987, the total number of winners since the beginning amounts to 2846, and an astounding 72.748 of total submissions (as of 2024) (Ars Electronica Linz GmbH & Co KG, 2022a). The awarded prize categories are switched, with some being awarded biannually instead of annually.

Each *Golden Nica* comes with a prize money of either 10.000€ or 5.000€, depending on the category. Besides the Golden Nicas, there also exists the *Award of Distinction* and the *Honorary Mention*. The trophy of the Golden Nica is based on the "[…] winged goddess Nike of Samothrace and was chosen as a trophy of the electronic and digital arts in 1986/1987 by Hannes Leopoldseder and Christine Schöpf together with the

graphic artist Johann Schorn" (Ars Electronica Linz GmbH & Co KG, 2010a). The word *NICA* was chosen based on the last syllable of "Ars Electronica", and the Greek goddess of victory, Nike.

All this makes the Ars Electronica one of the world's top addresses for media art. Necessary for this is that "both artists and scientists need to recognize, discuss, and ultimately confront the social and cultural phenomena that are the inevitable consequences of rampant technological change" (Grover, 2008, p. 54). Ars Electronica's goal from the beginning was to create an encompassing integration of art, technology and society in front of the constant backdrop of the future yet to come. They, as shown above, achieved this with the Prix Ars Electronica which is often credited with introducing media art to the world. Over the centuries, it has been proven countless times that technology clearly impacts society – often while the role artists play in this is still overlooked. Technological innovations emerge daily in academic, corporate and private sectors, yet they often lack a substantial societal impact - until effectively presented to the public. It is the constantly growing duty of artists and designers to adapt, package, and integrate new technologies into intuitive and elegant solutions for social issues. Only then can these innovations truly thrive and circulate throughout the general population.

All in all, one can see that the Ars Electronica truly has been a pioneer in the new media and digital art landscape, allowing up and coming artists, scientists and professionals of all other fields, to come together and merge their respective fields. The Archive and its connected database are necessary for this work to continue successfully, without which none of the artworks would be documented.

With the necessary background regarding the Ars Electronica established, this work will now proceed to its practical part: the case studies.

5 Case Studies

5.1 Introductory

One of the most important decisions to consider when it comes to a thesis analysing artworks is how to select suitable case studies. In general, they would most likely get selected based on art historical criteria, but this was not the case for this thesis since it is a master's thesis in the field of Digital Humanities with a heavy focus on the artistic status of software. The choice of original artworks for the case studies was functionally determined, based on four criteria by Falcão et al: contained, networked, user-dependent and generative artworks (Falcão et al., 2016). In a contained artwork, there exists no external influence on the software, with networked meaning the opposite: the functioning of the artwork depends on external data. In a user-dependent piece, the user "may change or distribute (parts of) the code or the actions of the user may be a combination of both" (Falcão et al., 2016). Regarding a generative artwork, the work is constantly being changed through software and can either be contained or dependent on external inputs.

These were the criteria for my four original chosen case studies, which I will now present very briefly. After this section, I will introduce the art pieces I ultimately chose and analysed.

The first artwork I had decided on was *TwiLight*, which I did analyse to a certain degree (more specific details on this later). *TwiLight* fit the criteria of a contained work, experiencing no external influence on the software.

The second artwork was called your *unerasable text* by Austrian artist Stefan Tiefengraber (Tiefengraber, 2011b). your *unerasable text* can be classified as an artwork of the second category – networked, in which the functioning of the piece depends on external data. In this work, participants were invited to engage in a simple task: sending a text message to a designated number written on a sign next to the installation (Tiefengraber, 2011a). The message was received by a mobile phone, which transferred the data to a computer. This data was consequently printed on an A6 sheet of paper, falling directly into a shredder. The message then remained readable for a few moments only to be destroyed seconds later; the shredded remains of each message form an ever bigger growing pile of paper on the floor, reminiscent of a generative graphic (Kunstuniversität Linz, 2012).

After I started analysing this piece, it became increasingly clear to me that it did not fit my desired criteria for a software-based artwork. Originally intrigued by the interplay

between the mobile phone, the printer, and the shredder, it did not display a clearly visible software-based part and rather fell under the category of an interactive art installation. This was not the focus of this thesis and led me to choose another piece in its place (more on this later).

The third artwork I had originally chosen was *Liquid Views*, created in 1992 by Monika Fleischmann and Wolfgang Strauss and produced at VISWIZ / GMD Gesellschaft für Mathematik und Datenverarbeitung (ISEA, 2022). It fell under the category of user-dependent work—the user(s) could change or distribute (parts of) the code, or the user's actions may be a combination of both.

In *Liquid Views*, the audience participated in an experience alike to Narcissus, behind a digitally activated mirror. A touchscreen with a mini camera simulated a water surface which reflected the viewer's image. Participants touching the screen led to the activation of algorithms that generated waves, which in turn distorted the viewer's reflections (ISEA, 2022). It examined themes like interactivity, artificial intelligence, Narcissus, neural networks and real time. *Liquid Views* fit the criteria for a software-based artwork better than your *unerasable text*. However, the software/algorithm source code was not accessible, which did not fulfil the expectations I had in mind when thinking of a user-dependent artwork which heavily relies on code.

The fourth piece was supposed to be *on your wrist is the universe*, by Adriana Knouf. *On your wrist is the universe* fell under the category of a generative software-based artwork, constantly changing through software, while being contained or dependent on external inputs (Falcão et al., 2016). The artwork could be described as a Pebble smartwatch, with the system's platform running a series of poems, which used real-time data regarding planets, satellites, rocket bodies and stars as the basis for generative poetry (Rhizome, 2016).

The project pushes the boundaries of electronic poetry – worn directly on the skin, the pieces become a tangible link to otherworldly experiences. A quick glance at one's wrist unlocks a poem, always close at hand.

On your wrist is the universe technically fit the criteria of a software-based artwork regarding accessible source code, but here again, it felt more like an interactive installation piece, caused by its heavy reliance on the hardware and less concentration on the software part.

The final work I had selected, specifically from the Ars Electronica Festival was *It Could Be You*. This piece was chosen without Falcão's specific criteria in mind and more based on the general feel and presentation while experiencing the artwork live during the festival. It incorporated a printer, thermal paper, software, real-time generative video and an installation variable (Cheng, 2024).

It Could Be You was described as a

> […] thought-provoking art project that explores privacy and identity in contemporary technology. Using machine learning and pentesting or ethical hacking, the project generates fictional personal data through synthetic data frameworks and, by gathering messages from various online forums and chatrooms, it raises questions about privacy and protection (Ars Electronica Linz GmbH & Co KG, 2023c).

The artist further elaborated that his work "[…] uses the nicknames of online real-time commenters as key elements for searching facial images. Through synthetic data and machine learning, it generates various faces and fabricated personal information, creating a scenario of 'It Could Be You'" (Cheng, 2024).

It Could Be You would have been the perfect case study if the source code had been openly accessible. Since the beginning I have wanted to portray and analyse an all-encompassing Ars Electronica artwork, which was unfortunately not possible with this one.

After starting with the analysis of the artworks and reviewing the state of selection of the case studies with my supervisor, we came to the unanimous conclusion that they did not fit the criteria for my thesis as I had imagined. This meant for me to re-select artworks to analyse, with the current status of my thesis and direction it was taking in mind. The initially set criteria inspired by Falcão did not make sense anymore at this point, and I had to redefine and reconstruct criteria while researching new artworks and finding out how they fit into my thesis.

TwiLight still fit the criteria of a software-based artwork and, more importantly, fit into the structure and body of my thesis at this point, which is why it was kept. The use case of *TwiLight* was one of an emulation, corresponding to the subchapter found in the section about the preservation of software-based art.

The second artwork I had re-selected resulted in *VR Tour through the Doomsday Clock* – my choice initially fell on it as I was looking for a substitute in place of your *unerasable text* (similar interaction between software and hardware, but with a stronger focus on the software aspect). After incorporating it into my thesis and experiencing it

side by side next to *TwiLight* the exciting dynamic between them started to emerge (this will be elaborated in more detail later) and the criteria of a VR artwork was born. I still wanted a specific Ars Electronica piece which ended up being re-selected last, since this pained the most to abandon – the final choice of my reselection fell on the *Unerasable Character Series* with its use case being software in its own, concrete infrastructure with fully accessible source code.

Based on this introduction, I will now present my three final case studies: *TwiLight*, *VR Tour through the Doomsday Clock* and the *Unerasable Character Series*. Each case study will be structured similarly – the artwork will be briefly introduced, followed by a short portrayal of the artist. After this, the previous exhibition history will be summarised, leading to the statement of significance. This chapter is split up into the artistic/historic significance and the technical/research significance. The artwork's significant properties will be explored later, culminating in a risk assessment and the final chapter on long-term risks, recoverability and preservation strategies.

As a reminder:

The best way to ensure that files will retain their value and use over time is to "[…] prepare and deposit the file formats with the highest probability of long-term preservation" (Mortimore, 2015). When the file formats possess the following characteristics, their likelihood for a successful long-term preservation of content is much higher. Since non-text format files (video files, spreadsheets and databases, images, audio and software) are especially vulnerable, additional precautions need to be taken in their regard for a successful and long-lasting preservation. Mortimore and Johns list seven criteria for submitted formats which heighten the likelihood of long-term preservation of functionality and content (Johns, 2025; Mortimore, 2015):

- complete and open documentation
- platform-independence
- non-proprietary (vendor-independent)
- no "lossy" or proprietary compression
- no embedded files, programs or scripts
- no full or partial encryption
- no password protection

File formats can generally be organised by their probability of long-term preservation of content and functionality. There are three different levels of probability: high, medium, and low. Mortimore and Johns recommend that scientists or research personnel upload

their works to a repository in the following formats and convert file formats with a lower probability of long-term preservation to a format with a higher likelihood (Johns, 2025; Mortimore, 2015).

Based on these criteria, I have decided to categorise each file from each case study's artwork. High-probability (A) file formats will be coloured **green**, medium-probability (B) in **yellow** and low-probability (C) in **red**. With this in mind, we can now proceed to observe the three case studies in their entirety, which form the last chapter of this thesis.

5.2 1st case study: *TwiLight*

I will now present my first case study, *TwiLight* which fits the use case of an emulation environment.

5.2.1 Description

TwiLight was a produced from 1991 to 1997 as a screensaver for ATARI TOS. It featured several different modules depicting scenes ranging from a penguin party to a swarm of butterflies or a deep-sea setting. The artists responsible for *TwiLight* are *Delirium Arts*, a German trio made up of Dragan Espenscheid, Alvar Freude and Peter Scheerer.

5.2.2 The Artists

Dragan Espenscheid is an 8-bit musician and media artist living and working in New York City. He has been leading the Digital Art Conservation Program at Rhizome since 2014 (Owens, 2014; Rhizome, n.d.).

Alvar Freude is a database specialist using Perl and/or PostgreSQL. His specialisations lie in performance optimisation and code quality. He currently heads the department for technical-organisational data protection and internet law at the State Commissioner for Data Protection and Freedom of Information Baden-Württemberg (Alvar Freude | republica, n.d.; Freude, 2002).

Finding current information about Peter Scheerer turned out to be impossible. He was responsible for the programming and design of *TwiLight*, but very little else about him was traceable (drx.a-blast, 2008).

5.2.3 Exhibitions

TwiLight has never been exhibited since it was programmed as a screensaver, and not with the intent of being an official artwork in mind. It was officially sold by Application Systems Heidelberg (Application Systems Heidelberg Software GmbH, 2025) up until 1997.

5.2.4 Statement of Significance

c. Artistic / Historic Significance

They released several modules throughout the years, “including Bat-O-Phants, North Sea 2000, Penguin Party, Ballonx, Butterflies, The Insane Cook, and Five Mutants” (Espenschied, 2012). The development of *TwiLight* has stopped, and it is now available to download as freeware; a few features include the extremely low memory usage through 100% assembler code and allegedly great compatibility with all types of applications (although it is questionable if this point still stands for today’s newer operating systems). A more specific feature is the “Simple but feature packed interface to C and assembler to write own modules. For example TwiLight gives you a color depth transparent sprite engine and the ability to play sampled sounds on any system” (drx.a-blast, 2012) and “Many options to adjust, for example *TwiLight* could be told to react different with certain applications in memory or listen to hardware ports” (drx.a-blast, 2012).

Graphics and animations were developed between 1991 and 1993 with Degas Elite and Cyberpaint II on an ATARI ST; 16 colours are the maximum any object can have, and objects that belong together have the same 16 colours, except for the explosion featured in the Five Mutants. The software runs on any ATARI TOS compatible system, including but not limited to TOS itself, MultiGEM, Geneva, MagiC, MagiC Mac, MagiC PC, and MultiTOS (Espenschied, 2012).

As mentioned before, seven additional modules have been released, which I will explain in further detail, supplied with screenshots, for the most accurate experience possible. Figure 2 depicts “Bat-O-Phants”, which features a classic pink Bat-O-Phant, although one could select from some colours. They were animated to fly from the right to the left side.

Figure 2: Bat-O-Phants

Figure 3 and 4 show the animation of “North Sea 2000” , which was used as a parody of the famous After Dark Aquarium, which had been the ruling screen saver application for Macintosh and Windows 3.11.

Figure 3: North Sea 2000

Figure 4: A screenshot of the animated North Sea 2000

“Penguin Party” (depicted in Figure 5 and 6) included an adjustable size of the penguin herd - what they were drinking (from water to ethanol), how much alcohol they could stomach and their thirst could all be configured. Depending on the individual settings, they would start to burp, vomit or drop, some sooner, some later. Sometimes an animated 16-ton weight would fall atop of one penguin, flatten the animal and then float back up (as illustrated in the two screenshots below).

Figure 5: Penguin Party

Figure 6: Penguin Party

Figure 7 is a screenshot of the “Ballonx” module, a relatively peaceful animation compared to the one seen before. At the bottom of the screen, a moving landscape can be seen, while some air vehicles are floating in the sky—two vintage-style airplanes, one hot-air balloon, and two airships make their way across the screen.

Figure 7: Ballonx

“Butterflies” (Figure 8) was based on monochrome handy scanner data, supplied by Markus Blank (Blank, 2008); the pictures were cleaned and coloured in Degas Elite.

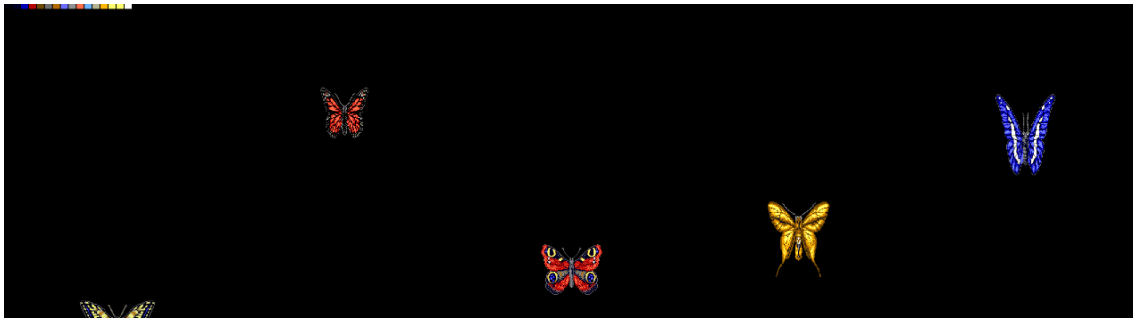

Figure 8: Butterflies

“The Insane Cook”, seen in Figure 9, featured a cook gone mad because his pans started to fly around, and he had to catch them. At the top left in the screenshot one unused colour is visible in the colour palette – it was originally planned that the user could change the sauce of the pancake by selecting a different colour in the colour palette, but the original graphics of pancakes with sauce unfortunately got lost

Figure 9: Insane Cook

The “Five Mutants” module, displayed in Figure 10, could be switched between two modes: in the first one, the mutants could be seen goofing around on the screen, in the second mode one could blast them with a crosshair. The explosion (in the bottom middle) was made in 1995 / 1996 with Apex Media on an ATARI Falcon 030. Some work was necessary on the animation of the original worm, to make it move forward in the right rhythm instead of appearing like sliding on an icy surface

Figure 10: Five Mutants

d. Technical / Research Significance

TwiLight included only a few, but for its time, significant features: it was greatly compatible with all types of applications and had many adjustable options, such as telling it to react differently depending on certain applications in memory. It had a very low memory usage thanks to its 100% assembler code, and combined with this, a simplistic but well-equipped interface to C and an assembler, allowing its users to write their own modules (drx.a-blast, 2012). Customising options included a colour depth engine and the possibility to play sampled sounds on any system.

5.2.5 Long-Term Risks, Recoverability and Preservation Strategies

e. Long-Term Risks

Emulating *TwiLight* was concluded as not successful, which I will now be illustrating on why by employing the emulation workflow previously established in chapter 3.7.4. Below the specifications for my host machine:
Processor: AMD Ryzen 7 5800H with Radeon Graphics - 3.20 GHz
Installed RAM: 16,0 GB (13,9 GB usable)
System type: 64-bit operating system, x64-based processor
Edition: Windows 11 Enterprise
Version: 23H2
OS build: 22631.4890
Serial number: YX02NJB0
Experience: Windows Feature Experience Pack 1000.22700.1067.0

***TwiLight* – Emulation Workflow:**

1. Emulator Selection

After identifying the specific processor architecture and operating system for which *TwiLight* was originally designed, I decided to attempt an emulation in SainT, my first guest system. This proved to be unsuccessful, as I was not able to open the AUTO_TL.PRG – file. This exact file was chosen after researching and finding it comparable to today's Windows .exe-files. When trying to open the .prg-file in the SainT emulator, the file could not be found in the AUTO-folder, and I was not given the option to access it through a different path.
A second emulation attempt was made in Gemulator after being unsuccessful in SainT: when trying to access the .prg-file, a message popped up reading "TwiLight wurde bereits installiert". The message was most likely in German, since *TwiLight* was created by the German artist trio "Delirium Arts".

As a last effort, a third emulator was tried—Hatari: the *TwiLight* folder itself did not show up in the GUI despite multiple attempts to restart the emulator. Only Diskstation A and Diskstation B were visible.

After all the aforementioned attempts had proven futile, the decision was made to contact Mr. Espenschied at Rhizome directly. Unfortunately, I never received a reply, so after waiting some time, I opted to contact three other email addresses found on the Rhizome website: the general information contact, Rhizome curatorial and Rhizome preservation. I received a reply from Michael Connor, the Co-Executive director at Rhizome. He informed me that *TwiLight* is currently "not in a restored state" (M. Connor, personal communication, September 9, 2024) and that "the issue may relate to the fact that TwiLight is a screensaver, and the software would configure the computer so that ti would come on when the computer is inactive" (M. Connor, personal communication, September 9, 2024). He further informed me that there is "a way to open the controll [sic] panel" (M. Connor, personal communication, September 9, 2024) and lastly asked me if I had tried accessing the .acc-file.

After failing to emulate *TwiLight* in the aforementioned emulators and reading up on the strengths and weaknesses of the different applications, I chose to try yet another software—Stella. Trying to access the .acc file in Stella led to receiving the following error message: "Unrecognized ROM type."

After again not being successful despite the input Mr. Connor had given me, I took to the internet and decided in a last attempt to use swarm-knowledge by posting in the official Atari forum and consulting professional Atari users.

I received three replies, which gave me a nudge in the right direction: I learned that ACC is not a ROM file but rather an accessory program that Atari TOS loads at startup - once it is loaded, it is accessible via the desktop. Stella is an emulator for 8-bit Atari machines, whereas *TwiLight* was designed as an accessory program for 16/32-bit Atari machines (which explains why Stella didn't recognise the ROM type).

For emulating 16/32-bit machines, I was recommended to use STeem, since it offers better support for Windows and works very straight forward coupled with TOS (The Operating System) version 1.62.

Over the next several days, I tried every possible sequence of steps to successfully emulate *TwiLight* with the now professionally recommended setup of emulator, TOS, RAM, colour, and resolution settings, but unfortunately, it still did not work.

I tried accessing the .acc file one last time since the emulator was wrong in the previous attempt, but the file was still not visible in the disk manager.

My other attempt was to drop the entire *TwiLight* .zip file to the A-Drive in the emulator, but STeem did not recognise any disk images. After affirming the contents' extraction to an ST drive, I received a pop-up about a successful extraction and was asked whether I would like to run STeem and go to the GEM desktop. Agreeing to this relaunched STeem, landing me on the desktop, with nothing happening, once again. This last unsuccessful attempt prompted me to try to emulate a completely different software to hopefully find out whether the error lay on my side or *TwiLight*'s side.

2. Disk Image Setup

I decided to emulate something completely different, a little more complex and fun than *TwiLight* and downloaded a disk image of the game "Mega Twins", a platformer popular from 1991 – 1993 (HOL TEAM, 1990; Sumpter & et al., 1993).

3. Emulator Configuration

Configuring the emulator settings for Mega Twins was no issue since all necessary information could be found online.

4. Emulator Execution

Running Mega Twins felt like child's play compared to *TwiLight* – I opened the appropriate disk (a .zip file, like *TwiLight*) in STeem and booted the disk image up, which instantaneously started the game successfully. – I had sound, working controls and a successful emulation running.

5. Emulation Assessment

Successfully emulating Mega Twins confirmed the suspicion I already had (and which was, at least partly confirmed by Mr. Connor in our email conversation) - *TwiLight* was indeed not in a state to be emulated, at least not by non-professional media conservation and restoration personnel.
This concluded my unsuccessful yet very enlightening attempt to emulate the Atari screensaver *TwiLight* as my first case study.

6. Emulation Documentation

No emulation available to document.

7. Emulation Archiving

No emulation available to archive.

File Formats

All the graphics and animations in *TwiLight* were created using Cyberpaint II and Degas Elite on an Atari ST; the software was intended for use on any Atari TOS compatible system such as MagiC, MagiC Mac, MagiC PC, TOS itself, Geneva and MultiGEM (Espenschied, 2012).

With *TwiLight* being the oldest piece of software in Rhizome's collection, four folders (AUTO, TL_SOUND, TL_SOURC.E, TLM) and an .**acc** file are revealed when downloading the folder to emulate the piece. „These ACC files are run with the help of the GUI called *GEM* (Graphical Environment Manager) which is a part of the Atari TOS (a derived version of TOS called the Atari TOS where TOS stands for 'The Operating System')" (FILExt., 2024). The ending *.acc* stands for (Desktop) Accessories, an essential software used in the Atari ST line-up.

Icon legend
first line in each cell = name of the artwork
files with ● = main folder
files with ○ = first subfolder
files with ■ = second subfolder
files with ❖ = third subfolder

• AUTO ○ AUTO_TL.PRG	• TL_SOUND ○ LIESMICH.TXT ○ BOING.SMP ○ BURP.SMP ○ CONEBLOW.SMP ○ DOORBL.SMP ○ EXPLODE.SMP ○ FINGER.SMP ○ FRUESTCK.SMP ○ GESTOERT.SMP ○ HICKS.SMP ○ HUI.SMP ○ KLO.SMP ○ PING.SMP ○ SCHRECK.SMP ○ TORÖÖ.SMP ○ TRAUM.SMP ○ TUSCH.SMP ○ WACH_AUF.SMP	• TL_SOURC.E ○ ASSEMBLE.R ▪ BEISPIEL.S ▪ ELEFANT.S ▪ LIESMICHT.TXT ○ NEU! ▪ TL_START.S ▪ TWILIGHT.H ○ PASCAL ▪ HEADER_P.S ▪ BILD_C.RSH ▪ SCHONPAS.PRJ ▪ SCHON_C.PRJ ▪ PAS_BILD.PRJ ▪ BILD_PAS.PRJ ▪ VERSUCH.PAS ▪ TL_INTER.PAS ▪ PAS_BILD.PAS ▪ BILD_PAS.C ▪ README.1ST ○ THREADS ▪ TL_START.S ▪ TL_DEBUG.S ▪ KONFIG.S ▪ TLM.PRJ ▪ DEBUG.PRJ ▪ DEMO.H ▪ DEMO.C ○ LIESMICH.TXT	• TLM ○ DATEN ▪ SPOT.GIF ▪ SPOTBAT.GIF ▪ HEADLINE.TXT ▪ BALLON.000 ▪ BREAKOUT.000 ▪ BUBBLES.000 ▪ DOSENSIM.000 ▪ ELEFANT.000 ▪ FALTER.000 ▪ FISCHE.000 ▪ KOCH.000 ▪ MONSTER.000 ▪ PINGU.000 ▪ PONG.000 ▪ UHR.000 ▪ BREAKOUT.001 ▪ BUBBLES.001 ▪ DOSENSIM.001 ▪ FISCHE.001 ▪ KOCH.001 ▪ MONSTER.001 ▪ PINGU.001 ▪ PONG.001 ▪ BUBBLES.002 ▪ DOSENSIM.002 ▪ FISCHE.002 ▪ KOCH.002 ▪ MONSTER.002 ▪ PINGU.002 ▪ PONG.002 ▪ DOSENSIM.003 ▪ PINGU.003 ▪ PINGU.004 ▪ PINGU.005 ▪ PINGU.006 ▪ PINGU.007 ▪ PINGU.008 ▪ DAWN ❖ BITBYBIT.BDI ❖ BUSYBITS.BDI ❖ FAEHRE2.BDI ❖ FLYFLOPY.BDI ❖ BITBYBIT.IMG ❖ BUSYBITS.IMG ❖ FAEHRE2.IMG ❖ FLYFLOPY.IMG ○ INF ▪ KONFIGS.INF ○ OVL ▪ RATZO.OVL ▪ SPRITE.OVL ▪ ST_SOUND.OVL ▪ THREAD.OVL	• TWILIGHT.ACC

Table 1: TwiLight files

Table 1 shows all the project files used in *TwiLight*. -

AUTO, the first main folder, contains a *.prg* file, an executable program designed for the Atari ST (Bitberry Software Aps, 2010c).

The second main folder, **TL_SOUND**, is made up of 18 files – 17 .smp files and one .txt file. *.smp* files are associated with ten different uses of the same .smp file extension, the most prevalent being the SmartMusic Performance File format (Geater, 2023). This is also the case for *TwiLight*, which makes the here mentioned .smp files a type of audio file. *.txt* files are traditionally used for "[…] simple plain text files, […] various text logs and reports many programs generate" (File-Extensions.org, 2025f).

The third main folder, **TL_SOURCE.E**, contains four subfolders and one README .txt file. The first subfolder, ASSEMBLE.R, contains one README .txt file and two files with the extension .S. These files can be described as modules for the assembler and do not have any real use standing alone (this was once revealed to me in a dream, 2024).

In the second subfolder *NEU!* are two files—again, one .S file and one .H file of the type C Header Source File. Files of this type facilitate working with resources since the search for object numbers can be omitted (Bitberry Software Aps, 2010b; ST Computer 05, 1987).

In the third subfolder, *PASCAL*, various file types exist – 11 files altogether, made up of one .S file, one .RSH file, four .PRJ files, three .PAS files, one .C file and one .1ST file. The *.RSH* file is most likely used to store the textures of the screensaver, with .RSH files most commonly known for the game Warhammer 40,000 containing game textures (File-Extensions.org, 2024f). Files with the *.PRJ* endings are usually responsible for storing project files (File-Extensions.org, 2024d); in the case of *TwiLight* they most likely contain the development notes, project settings, positions of saved windows and other raw data, which will not be visible in the final file. *.PAS* files relate to the programming language Pascal (in the case of *TwiLight PurePascal*) and mainly stood for source files when they were still in regular use (Dell, 2009). Opening the single *.C* file in a simple text editor reveals the line "actual C source, that compiles an ICON file" (Delirium Arts, 1991), making a relation to C Source Code the most logical one. The last file type in this folder is a .1ST file - *.1ST* files are most commonly associated with text files, distributed with applications, storing instructions and information for a software (File-Extensions.org, 2024a).

The fourth and last subfolder *THREADS* contains seven files: three .S files, two .PRJ files, one .H file and one .C file – all the file types mentioned here have been explained above – please refer to the appropriate sections.
The fourth and last main folder, **TLM**, contains 89 individual files and four folders. – *DATEN*, with its single subfolder *DAWN*, *INF* and *OVL*.
39 of the 89 files can be found in the *TLM* folder itself, with one .txt file and 38 .TLM files. After thorough research without a conclusive result, I strongly suspect that the ending .*TLM* refers to "*TwiLight* Modules" and is specific to this software. This conclusion was reached after discovering a thread regarding Atari screensavers on an Atari forum. In one reply, a user mentioned *TwiLight* as his preferred source of screensavers and referred to the TLM folder as the "module folder" and its content as "modules" (Mathias, 2008), which makes the abbreviation TLM plausible.
TLM's subfolder DATEN contains one subfolder and 45 singular files—12 files of the type *.000*, eight files ending with *.001*, seven files with the extension *.002*, two files ending with *.003*, and one file each following the same 00 pattern ending with the numbers from four to eight. There are also two .gif files and one .txt file.
The file endings from .000 up to .008 most likely refer to split archive/disk image files or backup archives (superuser1931, 2009).
The files inside of *DAWN* are split up into four .BDI and four .IMG files. The *.BDI* ending usually stands for Backup Data Image, storing a snapshot of data for backup or archiving during project development.
TLM's second subfolder *INF* contains one .INF file, necessary for Setup Information.
In *TLM's* last subfolder, *OVL*, four .OVL files are to be found. The *.OVL* extension hints at a relation to a DOS overlay file. These overlay files were necessary for managing memory limitations, which often came with running (now) old 640K MS-DOS machines: all program code often simply did not fit into the available memory at the same time. One solution for large programs to overcome this issue was the use of overlays. - "An overlay is a portion of the program code that shares memory for its code with other code modules. (part of [the] program to be loaded when needed), program code for DOS" (File-Extensions.org, 2024c).
In addition to the project files and folders described above, there are additional .gif files at one's disposal. Each separate GIF was split up into its various moving parts (different swimming fish, moving seaweed) which were individually embedded and layered onto one another on the website and as such could not be downloaded as one complete GIF, but only as their individually divided elements. In order to depict the complete GIFs as

accurately as possible, short screen recordings were taken of all of them, and static screenshots were captured to insert in this thesis in order to preserve the artwork in this shape.

f. Recoverability and Preservation Strategies

- AUTO
  - AUTO_TL.PRG
- TL_SOUND
  - LIESMICH.TXT
  - BOING.SMP
  - BURP.SMP
  - CONEBLOW.SMP
  - DOORBL.SMP
  - EXPLODE.SMP
  - FINGER.SMP
  - FRUESTCK.SMP
  - GESTOERT.SMP
  - HICKS.SMP
  - HUI.SMP
  - KLO.SMP
  - PING.SMP
  - SCHRECK.SMP
  - TORÖÖ.SMP
  - TRAUM.SMP
  - TUSCH.SMP
  - WACH_AUF.SMP
- TL_SOURC.E
  - ASSEMBLE.R
    - BEISPIEL.S
    - ELEFANT.S
    - LIESMICHT.TXT
  - NEU!
    - TL_START.S
    - TWILIGHT.H
  - PASCAL
    - HEADER_P.S
    - BILD_C.RSH
    - SCHONPAS.PRJ
    - SCHON_C.PRJ
    - PAS_BILD.PRJ
    - BILD_PAS.PRJ
    - VERSUCH.PAS
    - TL_INTER.PAS
    - PAS_BILD.PAS
    - BILD_PAS.C
    - README.1ST
  - THREADS
    - TL_START.S
    - TL_DEBUG.S
    - KONFIG.S
    - TLM.PRJ
    - DEBUG.PRJ
    - DEMO.H
    - DEMO.C
  - LIESMICH.TXT
- TLM
  - DATEN
    - SPOT.GIF
    - SPOTBAT.GIF
    - HEADLINE.TXT
    - BALLON.000
    - BREAKOUT.000
    - BUBBLES.000
    - DOSENSIM.000
    - ELEFANT.000
    - FALTER.000
    - FISCHE.000
    - KOCH.000
    - MONSTER.000
    - PINGU.000
    - PONG.000
    - UHR.000
    - BREAKOUT.001
    - BUBBLES.001
    - DOSENSIM.001
    - FISCHE.001
    - KOCH.001
    - MONSTER.001
    - PINGU.001
    - PONG.001
    - BUBBLES.002
    - DOSENSIM.002
    - FISCHE.002
    - KOCH.002
    - MONSTER.002
    - PINGU.002
    - PONG.002
    - DOSENSIM.003
    - PINGU.003
    - PINGU.004
    - PINGU.005
    - PINGU.006
    - PINGU.007
    - PINGU.008
    - DAWN
      - BITBYBIT.BDI
      - BUSYBITS.BDI
      - FAEHRE2.BDI
      - FLYFLOPY.BDI
      - BITBYBIT.IMG
      - BUSYBITS.IMG
      - FAEHRE2.IMG
      - FLYFLOPY.IMG
- TWILIGHT.ACC

			○ INF ▪ KONFIGS.INF ○ OVL ▪ RATZO.OVL ▪ SPRITE.OVL ▪ ST_SOUND.OVL ▪ THREAD.OVL	

Table 2: TwiLight analysed

TwiLight and its content were simultaneously the most complex and most straightforward files to categorise. Most complex, because the systems being able to run Atari TOS have long been out of order and are therefore difficult to access and/or to emulate, especially with regard to emulating a screensaver. For the easy aspect of the categorisation, the exact same arguments can be used. Since the system is so dated and the emulation was not successful (as proven by myself and further reinforced by Mr. Connor's email), nearly every file can be categorised as having a low probability of long-term preservation. Since the file type is so dated and additionally intended for an Atari machine, recommending an alternative better file type is unfortunately not possible here. The only high probability files were two .txt files (these can be left alone), which I was indeed able to access and two .gif files, which belong to category B – these should be converted to uncompressed .tiff files, for longer preservation success. A successful emulation still should be doable in some way, if attempted by professional conservation personnel with the proper resources – a legacy Atari machine, for which *TwiLight* was originally intended would ameliorate future emulation attempts noticeably.

For my second artwork, I ended up choosing *VR Tour through the Doomsday Clock* which is part of the show "It's Two Minutes to Midnight" by American new media artist Ellen Sandor and (art)^{n} artists Azadeh Gholizadeh and Diana Torres, inspired by Martyl [4]in collaboration with The Emerging Analytic Center at University of Arkansas at Little Rock and Carolina Cruz-Neira.

[4] Suzanne Martyl Langsdorf was best known as the designer of the Doomsday Clock, a prominent symbol for the possible destruction of nuclear weapons and the apocalypse (Langsdorf, 2015).

5.3 2nd case study: *VR Tour through the Doomsday Clock*

5.3.1 Description

The Archive of Digital Art describes the artwork (depicted in Figure 11) as

> […] being a part of the show "It's Two Minutes to Midnight," [it] provides viewers with an educational journey on humankind's history of de- and re-nuclearization. The show, organized by Weinberg/Newton Gallery in collaboration with the Bulletin of the Atomic Scientists in 2018, highlighted contemporary threats of nuclear weaponry, the climate change and such scientific discoveries as CRISPR genomic editing. The Bulletin, opened by the former Manhattan Project scientists in 1945, is a nonprofit organization which deals with global security threats deriving from the technological development. The organization keeps internationally known Doomsday Clock, a symbol of possible catastrophe induced by human activities. The VR tour navigates the spectator through the scenery based on aerial photography of Los Alamos, the area where the first atomic bomb was produced. The narrator, Rachel Bronson from the Bulletin of the Atomic Scientists, tells the history of the Doomsday Clock timeline from 1947 to 2018 (ADA, 2018).

Figure 11: Station 1968; VR Tour through the Doomsday Clock

With this in mind, the structure of this chapter will be explained briefly: the artist herself, Ellen Sandor, and past exhibitions of her work will be introduced. Following this, I will break down the artistic and historic significance of the artwork, supplemented with a number of screenshots of the different stations of the VR Tour. Since embedding animated gifs or videos is not possible in a text document, the supplied static screenshots fulfil more than the simple purpose of a one-dimensional graphics – they represent, for one, the closest approximation to experiencing the VR Tour in a text document possible, and two, their format is part of chapter 2.7 ("Long Term Risks, Recoverability and Preservation Strategies").
After the artistic and historic significance has been explained, I will be elaborating on the technical and research significance of the artwork – split up into three parts: the "CAVE-in-a-box"-system, (art)n PHSColograms and (art)n Virtual Reality. The significant properties of the work will be explained afterwards, with a risk assessment and long-term risks, recoverability and preservation strategies finishing up this case study.

5.3.2 The Artist

Ellen Sandor works as a new media artist and is the Founder/Director of the collaborative artists' group, $(art)^n$. She graduated with an MFA in Sculpture from the School of the Art Institute of Chicago in 1975; her studies introduced her to the exploration of the interplay between sculpture, video art and photography, all coupled with the inspiration she received from Outsider Art, also known as Art brut. Starting in the early 1980s, she had the "[…] unique vision to integrate these elements with nascent art forms including computer graphics, resulting in a new medium she called PHSColograms – 3D barrier-screen computer-generated photographs and sculptures" ((art)n, 2022), which can be described as the "Daguerreotypes of Virtual Reality". This has led to $(art)^n$ achieving significant breakthroughs in STEM and art, next to art history, tolerance and visual history. The $(art)^n$ group mainly works with PHSColograms, using them as backlit digital, immersive 3D photos, PHSCologram sculpture installations and related VR experiences ((art)n, 2022).

5.3.3 Exhibitions

VR Tour through the Doomsday Clock has been exhibited in the following context:

- It's Two Minutes to Midnight" (curator: Ellen Sandor, $(art)^n$ artists Diana Torres and Azadeh Gholizadeh), Weinberg/Newton Gallery, Chicago, 2018 [5]

5.3.4 Statement of Significance

g. Artistic / Historic Significance

The theme of this artwork is rather grim – albeit it having been created in 2018, the topics are still relevant today, if not even more than ever. It highlights the threats of nuclear warfare, constantly growing tensions between nations, and climate change, all while shining a light on groundbreaking scientific discoveries, like CRISPR genomic editing, which could have lifesaving effects in fields like healthcare and other applications (Gholizadeh et al., 2018). Looking back at this artwork now, in 2025, 7 years after its first exhibition, not much has changed – the Doomsday Clock currently stands at 89 seconds before midnight (last checked on March 1, 2025).

The exhibition featured a body of works that were all connected. The virtual reality piece on which my focus lies features textures and backdrops "[…] from Martyl's landscape paintings from *Mountain & Islands*, the *Doomsday Clock*, and archived concerns from the *Bulletin of the Atomic Scientists*" (Gholizadeh et al., p.1, 2018). *VR Tour through the Doomsday Clock* was displayed in a CAVE-in-a-box (cave automatic virtual environment) and in two Oculus Rift Stations.

Before I can elaborate on *VR Tour through the Doomsday Clock*, two other artworks need to be explained first, since these prior pieces were used as the basis for the virtual reality experience *Have a Nice Day* (Sandor et al., 2002) was initially inspired by Martyl's *Tent Rocks*, depicted in stark contrast with Martyl's *Doomsday* Clock, which was formerly designed in 1947 as the *Bulletin of the Atomic Scientists'* first magazine cover.

[5] This exhibition was documented and can be seen here: https://www.artn.com/it-is-two-minutes-to-midnight/.

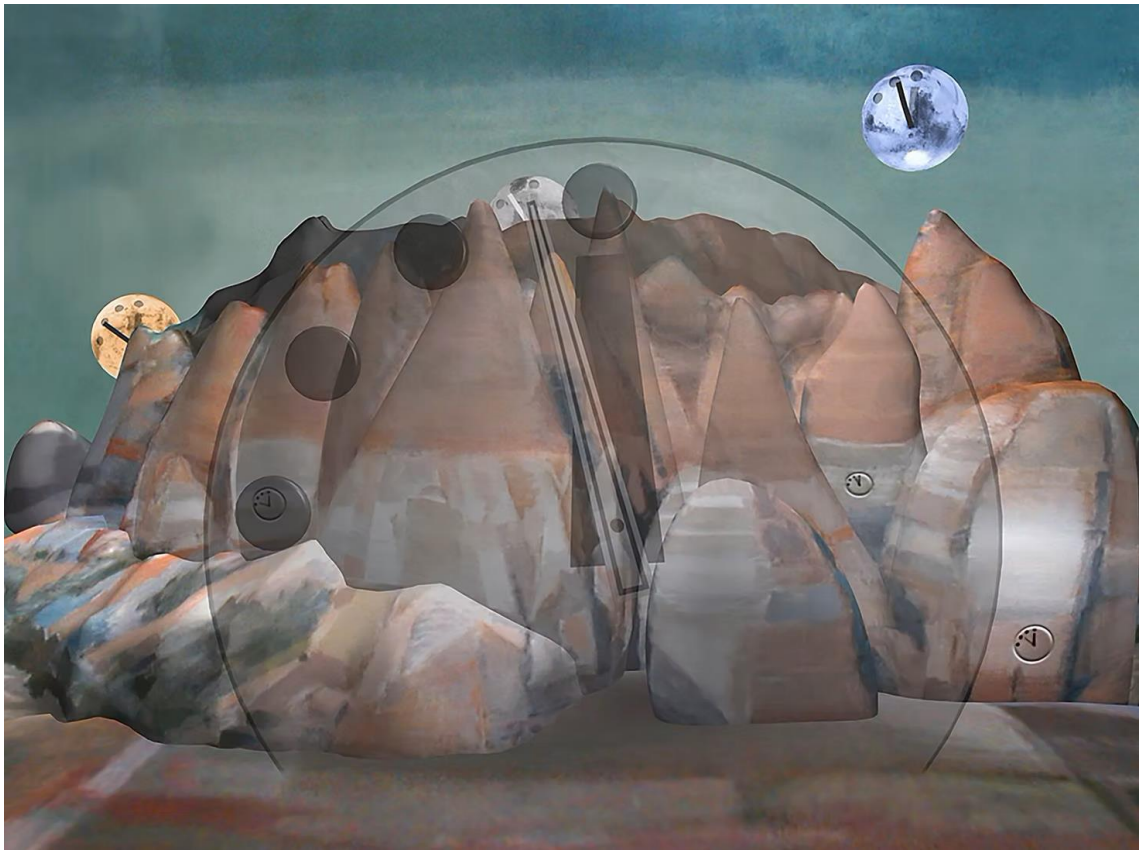

Figure 12: Have a Nice Day, 2002, Sandor et al.

The Clock (Figure 12), instantly recognisable, symbolically stands for the nuclear arms race and existential challenges created by humans, shown by scientific advancements. It has been relevant in 2002 directly after its creation, it has been relevant in 2018 when it became part of the exhibition and it still stands relevant today, in 2024, "[…] with other nations now admitting to the possession of nuclear weapons and fears of nuclear terrorism and recent increased terrorist activity" (Gholizadeh et al., 2018, p.1).
The series *Have a Nice Day II PHSCologram Sculpture* (Sandor et al., 2017) was based on the original *Have A Nice Day* as an homage to the late Suzanne Martyl Langsdorf. Sandor and her team at $(art)^n$ designed a PHSCologram sculpture with three added panels, complementing the original piece, "[…] addressing more in depth factors that have led and continue to push humanity towards midnight" (Gholizadeh et al., 2018, p. 3) – the series is depicted below in Figure 13 – 16.

Figure 13: Have a Nice Day II PHSCologram Sculpture, 2017, Sandor et al.

Figure 14: Have a Nice Day II PHSCologram Sculpture, 2017, Sandor et al.

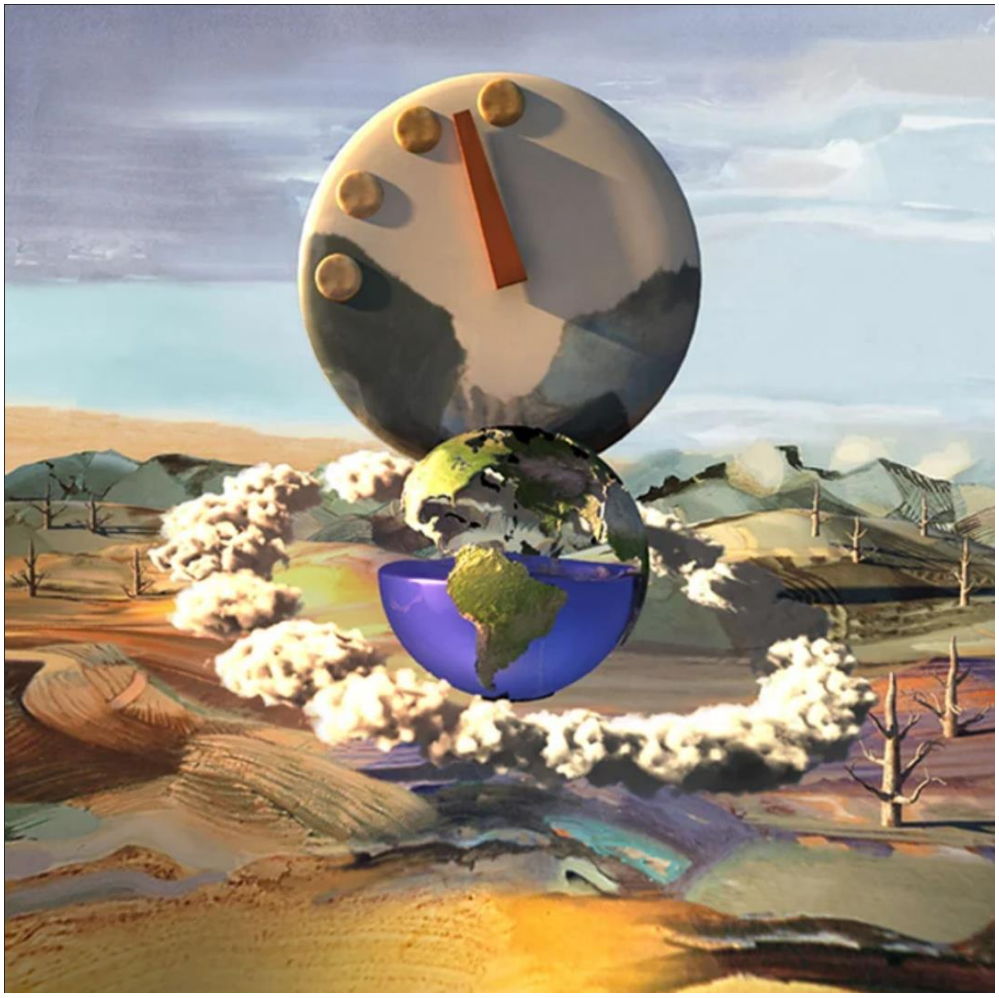

Figure 15: Have a Nice Day II PHSCologram Sculpture, 2017, Sandor et al.

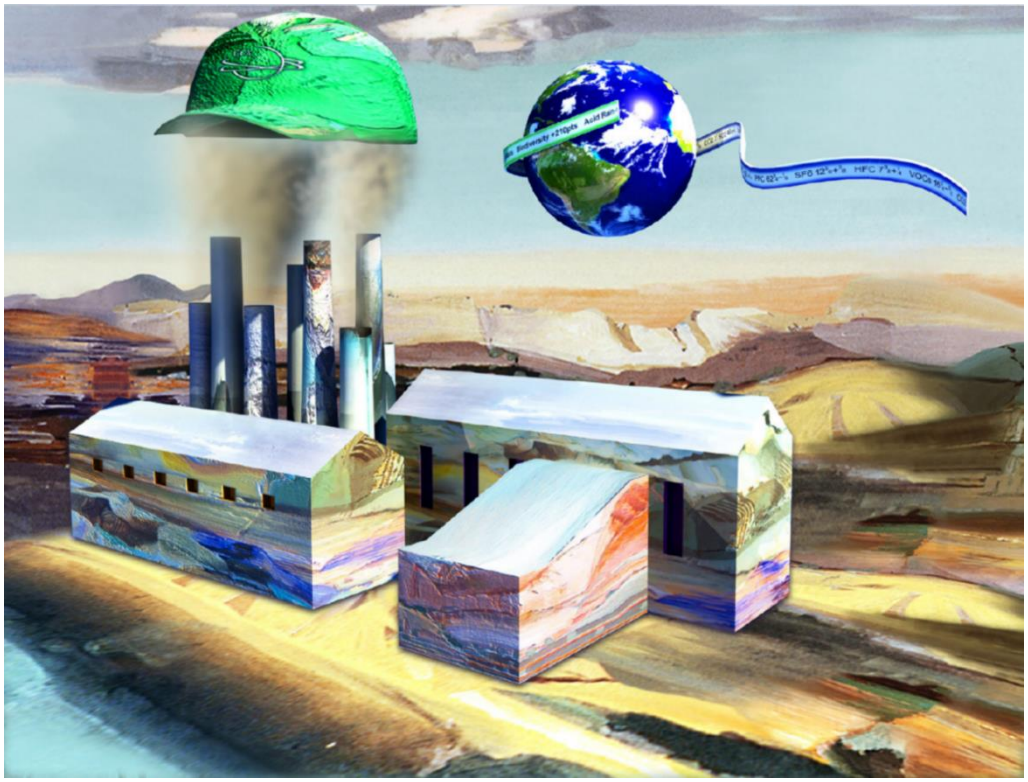

Figure 16: Have a Nice Day II PHSCologram Sculpture, 2017, Sandor et al.

Having explained the two previous artworks, I can now elaborate on *Have a Nice Day II: VR Tour through the Doomsday Clock* (Sandor et al., 2018), created from 1947 to 2018 and shown below as Figure 17. It was born out of collaboration between Martyl's paintings, Carolina Cruz-Neira, The Emerging Analytic Center, the University of Arkansas at Little Rock, Ellen Sandor, archived data from the Bulletin of the Atomic Scientists, (art)n's Diana Torres and Azadeh Gholizadeh (Wainwright et al., 2018, p. 68). – Together, they created a "[…] a unique VR experience tour that spans the history of the Doomsday Clock's activities" (Gholizadeh et al., 2018, p. 5). Having been produced 15 years after the original artwork I have illuminated beforehand, this artwork shines an even brighter light on the ever-growing dangers of nuclear warfare, heightened tensions between nations worldwide, and countless environmental factors of climate change, all accompanied by uplifting scientific discoveries harbouring the potential of improving medicine.

The piece is designed as a virtual landscape that the player can explore. It is situated in the desert site of Project Y, focusing on Los Alamos, navigating through the *Doomsday Clock* timeline from its creation in 1947 to 2018. Martyl's landscape paintings (all of the same location) were combined into a montage and used as the textures for the landscape; visual cues are placed at each station, symbolising significant events which have taken place in this specific year. A narration by Rachel Bronson, the CEO and president of the Bulletin of the Atomic Scientists (Gholizadeh et al., 2018, p. 6), is included.

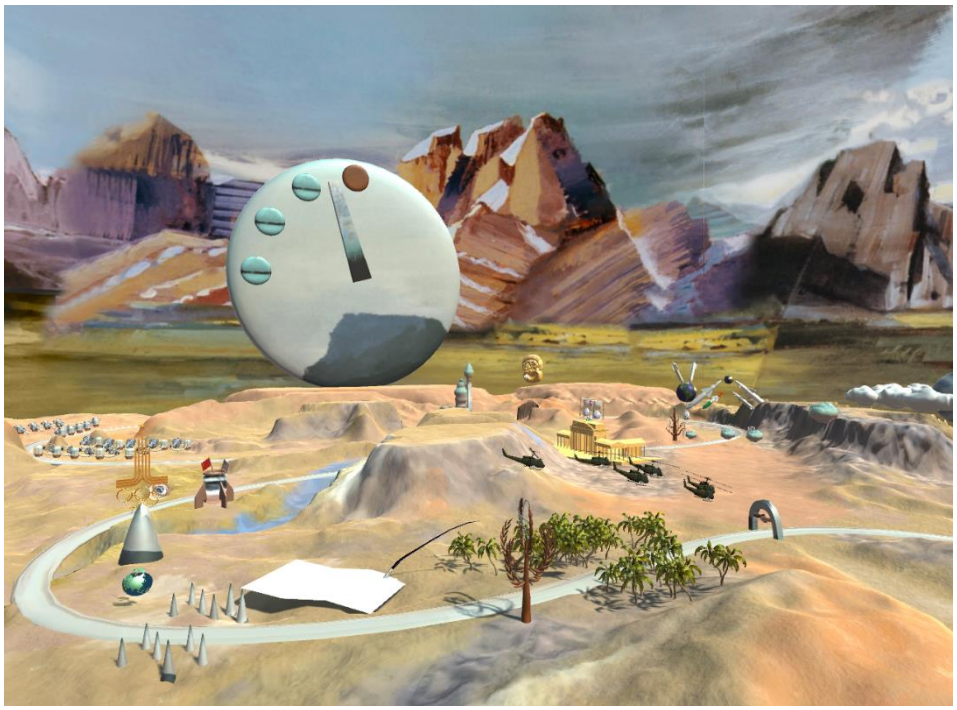

Figure 17: VR Tour Through the Doomsday Clock, 2018, Sandor et al.

I will now take a closer look at some of the stations in particular and their significance.

h. 1949: 3 Minutes to Midnight, The Arms Race is On:

Figure 18: Station 1949; VR Tour through the Doomsday Clock, 2017, Sandor et al.

Figure 18 - On 29 August 1949, the Soviet Union used a remote site in Kazakhstan to explode a nuclear device, very similar in design to the one used by the United States on Japan four years earlier. President Truman shares the news with the American public, albeit the Soviets denying the test, which marks the beginning of the arms race. The Bulletin of the Atomic Scientists explains

> We do not advise Americans that doomsday is near and that they can expect atomic bombs to start falling on their heads a month or year from now. But we think they have reason to be deeply alarmed and to be prepared for grave decisions. (Gholizadeh et al., 2018, p. 7)

In 1968, 19 years after the beginning of the arms race, the Eastern World is on fire –

i. 1968: 7 Minutes to Midnight, The Eastern World Explodes.

Figure 19: Station 1968; VR Tour through the Doomsday Clock, 2017, Sandor et al.

Simultaneously to the Cold War happening, the United States' entanglement in the Vietnam war intensifies (as depicted in Figure 19), Kashmir is fought over by India and Pakistan in 1965, and the Six-Day War in 1967 between Israel and its Arab neighbours has reignited the mutual hostile fire. Furthermore, China and France assert themselves as global players by developing nuclear weapons. This sequence of regular armed conflict paired with nuclear expansion leads to the Clock's minute hand being moved toward midnight. – "The Bulletin recognizes the possibility that these regional conflicts could flare into wider wars with the potential use of nuclear weapons" (Gholizadeh et al., 2018, p. 8).

In 1991, the Clock has been moved back 10 minutes and now stands at 17 minutes - this station is aptly named 1991: 17 Minutes to Midnight, A Fresh Start (visible in Figure 20).

j. 1991: 17 Minutes to Midnight, A Fresh Start

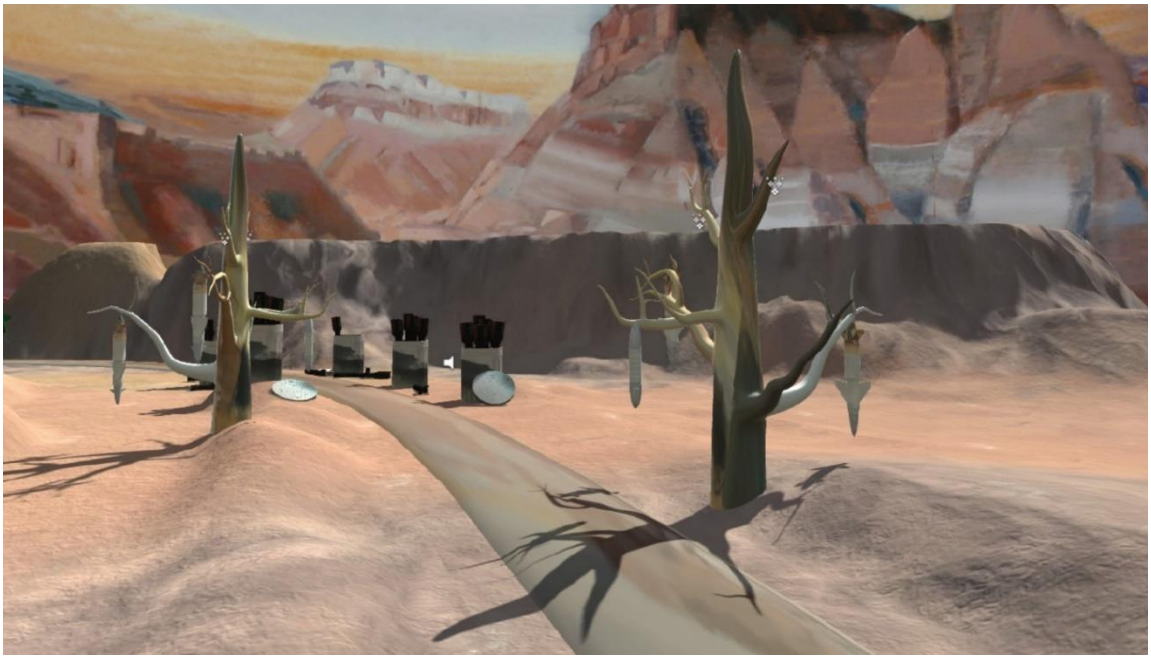

Figure 20: Station 1991; VR Tour through the Doomsday Clock, 2017, Sandor et al.

The Strategic Arms Reduction Treaty is signed by US President George H.W. Bush and Soviet President Mikhail Gorbachev, which marks the end of the Cold War and the beginning of them cutting back on nuclear arsenal. The agreement brings with it a considerable reduction of strategic weapons and, combined with a sequence of unilateral initiatives, a downgrade of the majority of bombers and missiles from the highest level of alert. “The illusion that tens of thousands of nuclear weapons are a guarantor of national security has been stripped away” (Gholizadeh et al., 2018, p. 8).

27 years later, in 2018, the clock is as close to midnight as it has never been before –

k. 2018: 2 Minutes to Midnight.

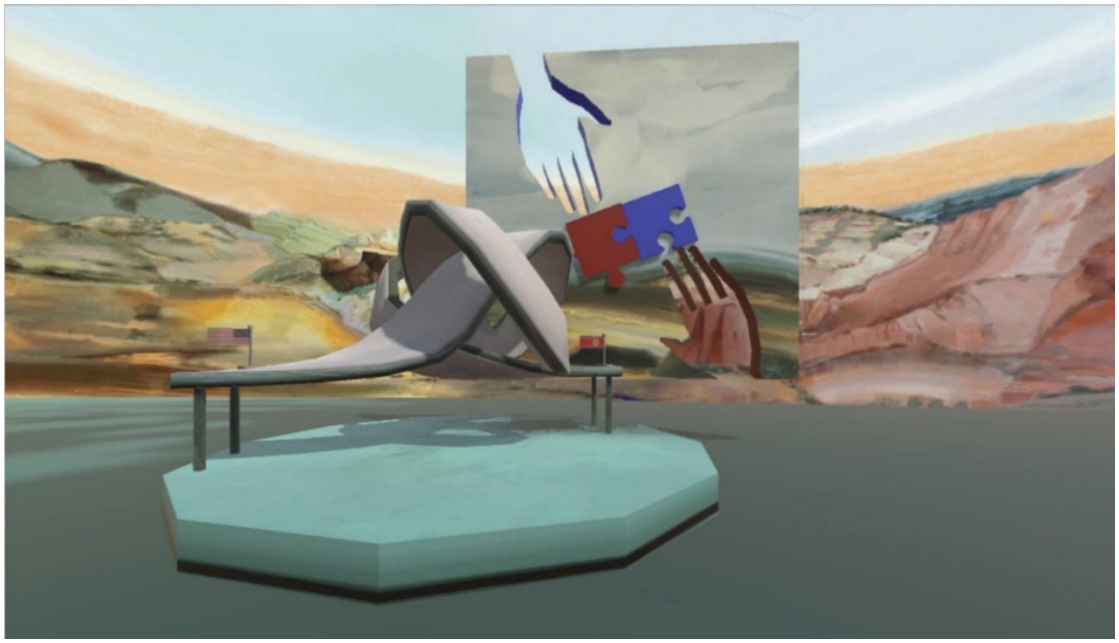

Figure 21: Station 2018; VR Tour through the Doomsday Clock, 2018, Sandor et al.

With world leaders spectacularly failing in an effective response to the ever-looming threats of climate change and nuclear war (Figure 21 – 25), the world's security situation is as dire as it was in 1953, the early days of the Cold War. North Korea has made remarkable progress in their nuclear weapons program, causing the nuclear realm to harbour the greatest risks for humanity once again, but specifically to themselves, other countries in the region and the United States (Gholizadeh et al., 2018, p. 9). "Hyperbolic rhetoric and provocative actions by both North Korea and the United States have increased the possibility of nuclear war by accident or miscalculation" (Gholizadeh et al., 2018, p. 10)

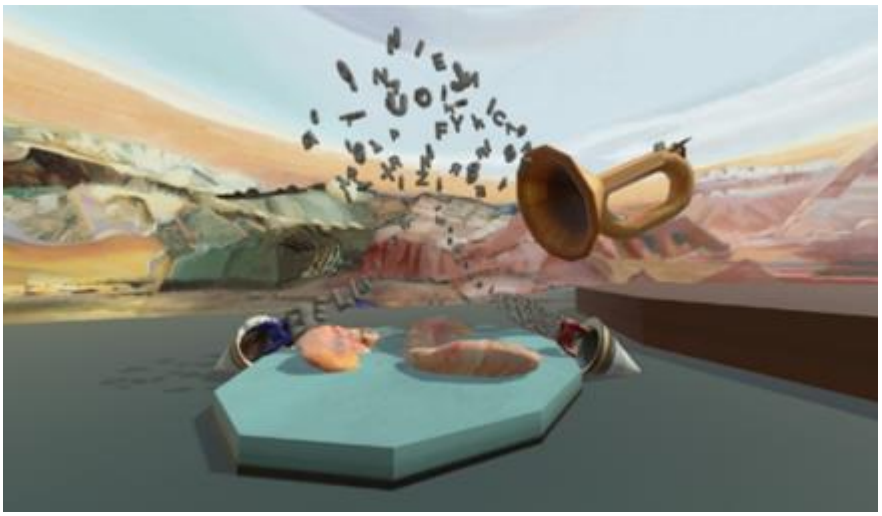

Figure 22: Station 2018; Reigning in Reckless Rhetoric, 2018, Sandor et al.

As if the problems occurring on the Korean Peninsula were not enough, Russia and the United States remain conflicted. NATO's borders are a training ground for military exercises, with both sides enhancing their nuclear armoury, making the future of arms control negotiations uncertain. In the Middle East and South Asia, the nuclear prospect appears similarly precarious.

Figure 23: Station 2018; Negotiate with North Korea, 2018, Sandor et al.

“The US questions the utility of the Joint Comprehensive Plan of Action, an internationally negotiated agreement to limit the military applications of Iran’s nuclear program” (Gholizadeh et al., 2018, p. 11). This causes panic among international partners, who are trying their utmost to preserve it, arguing for a safer world caused by the deal and insisting on improving it rather than turning a blind eye.

Figure 24: Station 2018; Stick with the Iran deal, 2018, Sandor et al.

Climate change-wise, the dangers may appear less immediate, but ignoring the steady, disastrous increase in temperature will require all the more attention when matters become so urgent, they cannot be avoided anymore (Gholizadeh et al., p. 11, 2018).

Figure 25: Station 2018; Insist on global action for climate change, 2018, Sandor et al.

5.3.5 Technical / Research Significance

l. CAVE-in-a-box

The CAVE, also called CAVE-in-a-box, was developed at the University of Arkansas at Little Rock and can be described as a "consumer social VR system". It is made up of a self-standing screen structure, creating an immersive room which can be occupied by five people simultaneously. With the structure being made of lightweight material it is easy to set up, disassemble and transport, "[…] making it ideal for projects that need to be experienced by small groups and for installations that need to travel to multiple locations, such as art shows, temporary museum installations, customers sites, and trade shows" (Gholizadeh et al., 2018, p. 20). The CAVE-in-a-box is designed specifically to "enable group virtual reality experiences" by cutting back on the need for VR - googles and -helmets, which limit social interaction of the users during the experience. – Users in the CAVE only need to wear 3D glasses, which facilitates any group virtual reality experiences, which makes it possible for the CAVE to be moved to different locations in order for an even larger audience to be reached (Gholizadeh et al., 2018, p. 20). With the CAVE being a single system, it is ready to be used "[…] 'off-the-box', as it comes with a high-end computer workstation and a developer's SDK (Software Development Kit) to create engaging social VR applications" (Gholizadeh et al., 2018,

p. 20). Since the SDK is compatible with existing SDKs of VR systems, applications previously used on other VR systems (helmets or googles) are easily portable to the CAVE-in-a-box. This can be described as a milestone in the field of VR by making social VR simpler, more affordable and truly usable for the broad public.

m. (art)n PHSCologrames

Ellen Sandor's professional background in Sculpture, with a deep interest in photography, video and Outsider Art led her to integrate these with other art forms, such as computer graphics, which ultimately resulted in the creation of the new PHSCologrames – "[…] 3D barrier-screen computer-generated photographs and sculptures" (Gholizadeh et al., 2018, p. 15). PHSCologram, pronounced as skol-o-gram, is a new media acronym comprising photography, holography, sculpture and computer graphics.

PHSCologrames (depicted in Figure 26) are created by digitally interleaving several rendered views of a virtual scene, "[…] in which the first line of every image is combined with the corresponding first line, and so forth until a recombined single image is made" (Gholizadeh et al., 2018, p. 21).

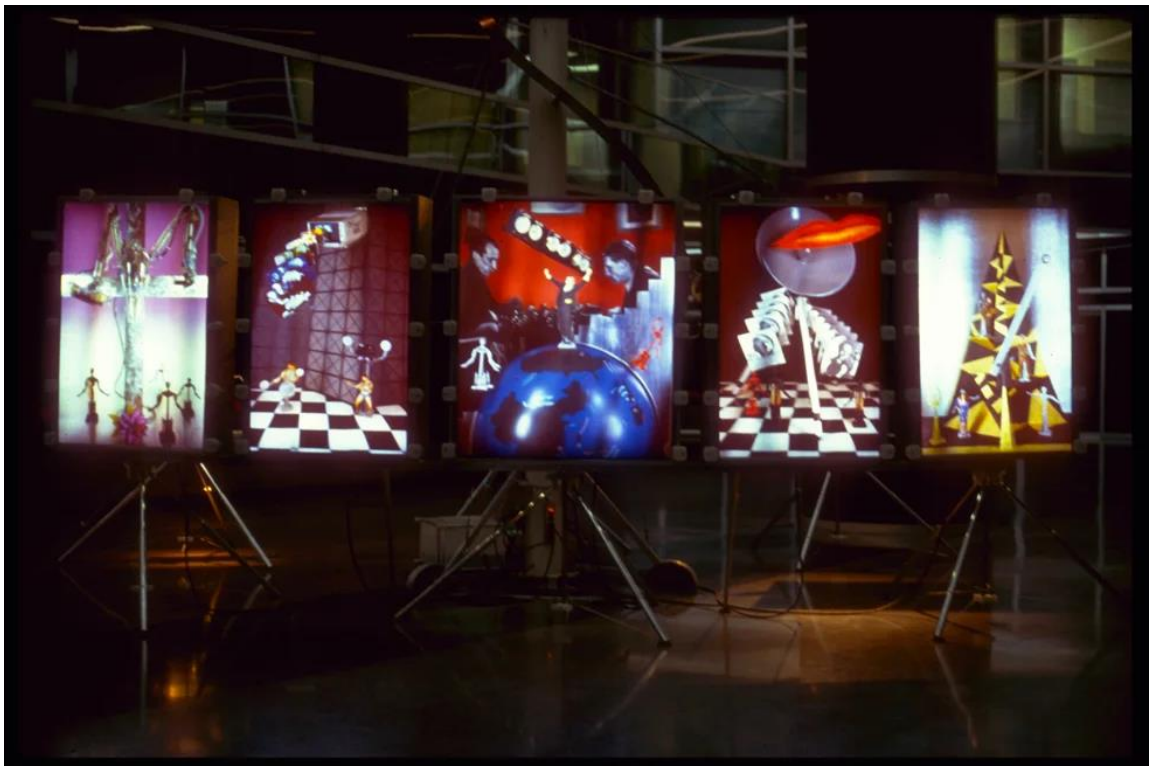

Figure 26: PHSCologram '83 by Ellen Sandor, Jim Zanzi, Mark Resch, Randy Johnson, and Gina Uhlmann, (art)n and Gary Justis, Jerry August, Tom Cvetkovich and Steven Smith (Sandor et al., 1984)

This resulting blurring of graphics into one singular piece is then connected to a line screen. A line screen can be described as a "[…] black piece of film with corresponding clear lines that is afixed to a piece of plexiglas, and allows a viewer to interpret the digital photograph as a three-dimensional sculptural object when backlit" (Gholizadeh et al., 2018, p. 21).

n. (art)^n Virtual Reality

(art)^n uses Autodesk Maya to create their virtual environments – a powerful three-dimensional computer graphics application, which the collective has been using for 20 years. The animations and creative three-dimensional images are crafted with the help of a powerful engine where complex modelling and rendering tools work together. The subsequent development of the virtual environments and programming is being achieved with the help of Unity 5, which is primarily used for game creation. Unity 5 "[…] provides a rich ecosystem that allows the artist to weave together nearly every form of digital asset–images, three-dimensional models, audio, and more – with a powerful object-oriented scripting language allowing for boundless creativity" (Gholizadeh et al., 2018, p. 21). The resulting virtual worlds created with Maya and Unity 5 can be interacted with and navigated through the use of the Oculus Rift; controlling these three-dimensional environments happens by the body's movement.

5.3.6 Long-Term Risks, Recoverability and Preservation Strategies

o. Long-Term Risks

Icon legend
first line in each cell = name of the artwork
files with ● = main folder
files with ○ = first subfolder
files with ■ = second subfolder
files with ❖ = third subfolder

Have A Nice Day	Have a Nice Day II PHSCologram Sculpture	Have A Nice Day II: VR Tour through the Doomsday Clock
• Have A Nice Day,png	• Have A Nice DayII.gif • Have A Nice DayII_2.gif • Have A Nice DayII.png • Have A Nice DayII_2.png • Have A Nice DayII_3.png • Have A Nice DayII_4.png	• Station 1949; VR Tour through the Doomsday Clock.png • Station 1968; VR Tour through the Doomsday Clock.png • Station 1991; VR Tour through the Doomsday Clock.png • Station 2018; Insist on global action for climate change, 2018.png • Station 2018; Negotiate with North Korea, 2018.png • Station 2018; Reigning in Reckless Rhetoric, 2018.png • Station 2018; Stick with the Iran deal, 2018.png • Station 2018; VR Tour through the Doomsday Clock.png • VR Tour through the Doomsday Clock.webp • Virtual Reality Tour Through the Doomsday Clock, Detail, 1968.mp4 • VR Tour through the Doomsday Clock, Detail, 2018-19.mp4

Table 3: It Is Two Minutes to Midnight files

This piece was the least accessible in terms of its specific project files. - None of the software files per se were openly available for download, as visible from Table 3. If they had been accessible, using the *Virtual Reality Artwork Acquisition Template*, as referenced in Chapter 3.4 would have been advisable. The only possibility available was to analyse the embedded videos, pictures and exhibition catalogue, which serve as documentation material for the artwork and do not constitute an integral software part of the artwork itself. – This makes their aspired likelihood for a successful long-term preservation all the greater, as mentioned before in Chapter 3.

The first available picture, *Have a Nice Day*, was taken from the first part of the artwork – a simple .png file taken as a screenshot from the exhibition catalogue.

For the second part of the artwork, *Have a Nice Day II PHSCologram Sculpture*, more documentary material was available – two GIFs on the website and four pictures in the exhibition catalogue. The formats here were two .gif files and four .png files, as these were taken as direct screenshots from the exhibition catalogue.

For the third and final part of the artwork, *Have A Nice Day II: VR Tour through the Doomsday Clock*, the most documentary material was available – eight .png files from

the exhibition catalogue (again taken as screenshots), a .webp file (as a download from the website) and two videos in .mp4 format.

.webp could be described as a problematic file extension – like .jpg, it is used as a file format for images, with better performance than .jpg files. Google originally created it, "[…] designed to improve the performance of web pages and make the web faster by reducing image file sizes" (File-Extensions.org, 2025g). Problems arise when .webp files are edited repeatedly – since the format is lossy, some data is lost during encoding, reducing the overall quality of the image (File-Extensions.org, 2025g).

mp4 files contain "[…] MPEG-4 encoded video and Advanced Audio Coding (AAC)-encoded audio content" (File-Extensions.org, 2025d) and can be called the "most common file extension used for the MPEG-4 format" (File-Extensions.org, 2025d). The videos, like the .webp file, could be viewed on the website and were recorded using a screen recorder. One video is accompanied by audio commentary, while the other is not (which causes me to perceive this as an incomplete upload). The automatic save format for screen recordings on Windows 11 is .mp4.

p. <u>Recoverability and Preservation Strategies</u>

Have A Nice Day	Have a Nice Day II PHSCologram Sculpture	Have A Nice Day II: VR Tour through the Doomsday Clock
• Have A Nice Day,png	• Have A Nice DayII.gif • Have A Nice DayII_2.gif • Have A Nice DayII.png • Have A Nice DayII_2.png • Have A Nice DayII_3.png • Have A Nice DayII_4.png	• Station 1949; VR Tour through the Doomsday Clock.png • Station 1968; VR Tour through the Doomsday Clock.png • Station 1991; VR Tour through the Doomsday Clock.png • Station 2018; Insist on global action for climate change, 2018.png • Station 2018; Negotiate with North Korea, 2018.png • Station 2018; Reigning in Reckless Rhetoric, 2018.png • Station 2018; Stick with the Iran deal, 2018.png • Station 2018; VR Tour through the Doomsday Clock.png • VR Tour through the Doomsday Clock.webp • Virtual Reality Tour Through the Doomsday Clock, Detail, 1968.mp4 • VR Tour through the Doomsday Clock, Detail, 2018-19.mp4

Table 4: It is Two Minutes to Midnight analysed

The second newest artwork, *It is Two Minutes to Midnight*, can be defined as relatively straightforward regarding its preservation. Since I do not have access to the original artwork files, I can only speak about the available documentation material regarding long-term preservation. From the 18 total files, 17 can be categorised as file formats with medium probability for long-term preservation - 12 of them are .jpg files, two are .gif files, and two are .mp4 files. The only file with a low probability for long-term preservation is a .webp file, which should be converted to an uncompressed.tiff, lossless .jpeg2000 or .png file.
Similar to the *Unerasable Character Series*, the suggested solution here would be to convert the endangered file formats to a higher probability format.

5.4 3rd Case Study: *Unerasable Character Series*

5.4.1 Description

This three-piece work addresses "the chilling scale and effect of state-enacted censorship, enforced through digital infrastructures" (Soon, 2023). It thematises the politics of erasure and the temporality of voices, all embedded within the context of digital authoritarianism. The series is constructed as a "lyrical repository" for oppressed voices, "with each technically scrutinizing and poetically portraying tweets censored from Weibo", one of China's most popular social media platforms. It encompasses the vast amount of silenced and unheard voices by "technically examining and culturally reflecting the endlessness" and broad consequences of censorship being technologically enforced through platforms and their infrastructure.
Data collected is made up of erased / censored textual data based on Weibo, via a system called "Weiboscope", a data collection and visualization project by Dr. King-wa Fu from the University of Hong Kong. The trained system has been sampling timelines of a chosen amount of Chinese microbloggers with more than 1.000 followers or whose posts are regularly deleted or otherwise censored.

5.4.2 The Artist

Winnie Soon, originally born and raised in Hong Kong, is a Senior Lecturer/Course Leader at the Creative Computing Institute at the University of the Arts London, with over 10 years of experience in teaching, academic research and leadership. They are the co-principal investigator in the Digital Activism research project, belonging to the Shaping Digital Democracy (SHAPE) research centre, which is funded by Aarhus University (Andersen, 2023; ual, n.d.) and also participate as a co-research lead in

British Digital Art, British Art Network (British Art Network, n.d.). Soon's research and practices focus on the intersection of technology and art, specifically in the areas of Software Studies and Computational Cultures, thereby working with topics such as digital censorship, minor technology and computational publishing "to understand the potential and implications of technology in a wider cultural and societal context" (ual, n.d.) and queer code and coding in general. Their works have been published and shown worldwide – at libraries, universities, museums and galleries, conferences, and art/science festivals, as well as in the United Kingdom, Germany, Austria, Denmark, Poland, Italy, and the Netherlands. In relation to these showings, they have won several significant art awards, including the Golden Nica for *Unerasable Character Series*, the WRO 2019 Media Art Biennale Award, the 17th and 26th ifva award and the Expanded Media Award for Network Culture at Stuttgarter Filmwinter (Soon, n.d.).

5.4.3 Exhibitions

Unerasable Character Series as a compilation has been exhibited once at the Ars Electronica Festival in 2023, and repeatedly in several different contexts as singular works:

- *Unerasable Character Series I* (Soon, 2022b):
  - 2024, Faraway So Close, ICC in Tokyo, Japan
  - 2024, Hello Human, MoCA, Taipei
  - 2023, Matsudo International Art Science Festival 2023, Japan
  - 2023, Ars Electronica, Austria (as a compilation with UCII and UCIII - Received the Golden Nica in the *Artificial Intelligence and Life Art* Category
  - 2022-2023, Data Relations, Australian Centre for Contemporary Art, Australia
- *Unerasable Character Series II* (Soon, 2020b)
  - Oct 2022, The Production of Post-Truth, Ugly Duck, London
  - May 2022, Living Protocols, Harvard Art Museums Lightbox Gallery, USA
  - Feb 2022, [esc] media art lab, Austria
  - Mar 2021, Posthuman exhibition as part of Electronic Literature Organization, University of Bergen, Norway. Curated by Eamon O'Kane, Jason Nelson, Scott Rettberg and Joseph Tabbi. (also available online)
  - Mar 2021, 26th ifva awards, Pao Galleries, Hong Kong Arts Centre. (Received the Special Mention Prize in the media art category)
- *Unerasable Character Series III* (Soon, 2021b):
  - May 2022, Newman Library, Virginia Tech, USA

- 2022, Metatopia Pavilion, The Wrong Biennale
- Feb 2022, Electronic Literature Directory by Brent Lace
- May 2021, Connecting the dots, curated by Joel Kwong, Hong Kong
- May 2021, The New River – A Journal of Digital Art and Literature (Spring 2021 issue)

5.4.4 Statement of Significance

q. Artistic / Historic Significance

The *Unerasable Character Series* addresses the chilling reach and effect of state-enacted censorship, further enforced by operating digital infrastructures. The three combined works "[…] act as lyrical repositories for suppressed voices, with each technically scrutinizing and poetically portraying tweets censored from Weibo, one of the biggest social media platforms in China" (Ars Electronica Linz GmbH & Co KG, 2023b). The work investigates the frail existence and temporality of voices embedded in the context of authoritarianism enacted through digitality and studies politics of erasure. Through technical inspection and cultural reflection, Soon portrays the absolute endlessness of the unheard voices, further investigating the broader consequences of censorship being enforced through technological infrastructures and policies.

r. Technical / Research Significance

The artwork series achieves all this by collecting unheard voices in the shape of erased or censored textual data, based on, as mentioned above, one of the biggest social media platforms in China – Weibo. The tool used for this is called the "Weiboscope", a project developed for data visualisation and collection by Dr. King-wa Fu from the University of Hong Kong (Soon, 2021b). Weiboscope has been collecting its data by frequently sampling timelines of a selected group of chosen Chinese microbloggers, who a) are regularly being censored in their posts or b) have more than 1000 followers (Ars Electronica Linz GmbH & Co KG, 2023b).

Unerasable Character Series I comprises two significant components related to Machine Learning: the first one being the input data, which was collected between 30 June 2021 and 30 June 2022. This collected data set made up 54.064 sample censored posts, amounting to a pile of more than 6000 pages. The second component embodied the predictive / output data from each iteration of the ML training process; this output data, based on the censored input text, was then turned into a DIY book with customised binding tools (Ars Electronica Linz GmbH & Co KG, 2023b). The book is intended to be generative yet uncomprehensible to avoid censorship, inspired by blank paper

revolutions and white paper protests (the book can also be defined as a Significant Property, where it will be mentioned again).

Unerasable Character Series II "[…] consists of a custom-software that scrapes the erased 'tweets' from Weiboscope on a daily basis, and the project presents the living archives in a grid format" (Ars Electronica Linz GmbH & Co KG, 2023b). The single tweets are deconstructed into a character-by-character display, where they are displayed for a certain amount of time. This duration of visibility is determined by the real amount of time the post was present and visible om Weibo before its removal. This way of displaying the posts alters the artwork "[…] from a busy cacophony of voices and characters to a silent and empty space, marked by the disappearance of the characters" (Ars Electronica Linz GmbH & Co KG, 2023b).

Unerasable Character Series III makes use of COVID-19-related data from 1 December 2019 to 27 February 2020, when the outbreak in China started. King-wa Fu and Yuner Zhu reported "[…] 11,362,502 posts during the period, among which 1,230, 353 contain at least an outbreak-related keyword and 2,104 (1.7 per 1,000) posts had been censored" (Ars Electronica Linz GmbH & Co KG, 2023b). UCIII shows all the erased archives in a web presentation, with each post being illegible by either being blacked out or otherwise obscured. The only characters remaining are the emojis, special characters and punctuation and the fuzzy timestamps and breaks, illustrating "affective and expressive, as well as temporal and spatial dimensions of unheard voices" (Ars Electronica Linz GmbH & Co KG, 2023b).

5.4.5 Long-Term Risks, Recoverability and Preservation Strategies

As previously referenced in Chapter 3, the *Unerasable Character Series* displays the most accessible software source code and, therefore, the most likely artwork to be preserved successfully.

s. Long-Term Risks

Icon legend
first line in each cell = name of the artwork
files with ● = main folder
files with ○ = first subfolder
files with ■ = second subfolder
files with ❖ = third subfolder

unerasable-characters-i-main	UnerasableCharactersII-master	unerasable-characters-iii-master
• hmtltoprint	• code	• DataProcessing
○ coverImg	○ assets	○ .gitkeep
▪ comverImg1.jpg	▪ HanyiSentyTang.ttf	○ cleaning_dividefile.py
○ font	○ Libraries	• visual
▪ GenRyuMinTW-M-01.ttf	▪ p5.js	○ .gitkeep
○ output.html	○ scrape	○ unerasablecharacters3.png
○ style.css	▪ weiboscope_scrape.py	○ unerasablecharacters3c.png
• .gitlab-ci.yml	○ index.html	○ unnerasablecharactersIII.gif
• LICENSE	○ silence.js	• WebPage
• README.md	○ sketch.js	○ assets
	• images	▪ .gitkeep
	○ installation1.png	▪ favicon.png
	○ installation2.png	○ .gitkeep
	• LICENSE	○ index.html
	• README.md	○ style.css
	• unersablecharactersII.gif	• README.md

Table 5: Unerasable Character Series files

The artwork *Unerasable Character Series* is divided into three parts, each written in a number of different languages (see chapter 3.8.2 for more information on source code annotation and preservation).

Unerasable Character Series I has been technically produced with Python, HTML, CSS and Paged.js (Soon, 2022b). HTML and CSS, as explained in subchapter 6.2 fall under the category of mark-up languages, while *Python*, as an interpreted language, can be defined as a "[…] dynamic object-oriented programming language" (File-Extensions.org, 2024e) and is used for a broad variety of software development. *HTML* (Hypertext Markup Language) is primarily known as the main language for writing web

pages, formatting tables, texts, images and other content which can then be interacted with on any web browser. It "[…] provides means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items" (File-Extensions.org, 2025b). The abbreviation *CSS* stands for Cascading Style Sheet, fulfilling the use of "[…] formatting the contents of web pages. It […] contains customized, global properties and other information about how to display HTML elements" (File-Extensions.org, 2025a) and can define a variety of characteristics, such as the background, colours, line spacing, fonts, size or location of HTML elements.

Paged.js is "[…] a free and open-source library that paginates any HTML content to produce beautiful print-ready PDF. The library fragments the content, reads your CSS print declarations and presents a paginated preview in your browser that you can save as PDF" (Coko, 2019).

When downloading the project file from Gitlab, the .zip file contains a folder (*htmltoprint*) and three individual files – a readme.md file, a usage license, and a *.yml* file – a template telling Gitlab how to automatically build, test, and start the HTML-based project when changes are made to the code. The file extension *.yml* is related to YAML (YAML Ain't Markup Language), a "[…] human-readable data serialization format" (File-Extensions.org, 2024h), taking concepts from languages such as Python, Perl, XML or C. *.md* stands for Markdown, as it is related, among others, to source codes and texts and Markdown markup language (File-Extensions.org, 2024b). It is further characterised as a "[…] lightweight markup language, to write using an easy-to-read, easy-to-write plain text format, then convert it to structurally valid XHTML (or HTML)" (File-Extensions.org, 2024b).

In the *htmltoprint* folder are two individual files (*style.css* and *output.html*) and two folders – *coverImg* with a cover photo (a .jpg file) for the project and *font* containing a TrueType font file. The commonly known file extension *.jpg* is "associated with JPEG, […] a lossy image compression algorithm that significantly reduces the file size of the original image at the cost of quality" (File-Extensions.org, 2025c); the compression ratio is relative to the quality of the .jpg file – the higher the ratio, the lower the quality. *.ttf*, a vector image format, stands for TrueType font – it is an "[…] outline font standard […] used to store a set of glyphs, characters, or symbols such as dingbats" (File-Extensions.org, 2024g).

Unerasable Character Series II was realised using Python and p5.js (Soon, 2020b). *p5.js* is a „[…] free and open-source JavaScript library built by an inclusive, nurturing

community. […] p5.js supports audio-visual, interactive, and experimental works for the web" (p5.js Contributors et al., 2021). In this .zip file, there are two folders (*code* and *images*) and again three individual files – a license, an .md file and a .gif file of the second artwork. The file extension *.gif* stands for Graphical Interchange Format, making it an image file. GIFs are most often used for website animations, web graphics or for the creation of graphic text for websites (Bitberry Software Aps, 2010a). In the *code* folder are three subfolders – *assets* (in it a .ttf file), *libraries* (with a p5.js file), *scrape* (containing the Python source file) and the three individual files *index.html*, *silence.js* and *sketch.js*. In the second main folder, *images*, there are two .png files – *installation1* and *installation2*. *.png* stands for Portable Network Graphics and is another type of image file, similar to .gif or .jpg. The important difference lies in it being a lossless format, making it superior to .jpg or .gif files regarding long-term preservation. It has also been approved of the WWW consortium as a standard to replace .gif as "[…] GIF uses a patented data compression algorithm called LZW. In contrast, PNG is entirely patent- and license-free" (File-Extensions.org, 2025e).

Unerasable Character Series III implemented the 3-month-long COVID-19-related data collection from Weibo in .csv (King-wa Fu & Yuner Zhu, 2020), Python for data parsing and HTML and CSS for web presentation (Soon, 2021b). The folder of the third artwork piece is divided into three main folders and a single .md file. The first folder, *DataProcessing*, contains a .gitkeep file and a Python Source File. In the second main folder, *visual*, there is one .gitkeep file, two .png files of the artwork, and one .gif file of the artwork. The last folder, *WebPage*, is divided into a subfolder, *assets*, and three individual files - .gitkeep, index.html, and style.css. A .png file and a .gitkeep file are stored in assets.

t. Recoverability and Preservation Strategies

unerasable-characters-i-main	UnerasableCharactersII-master	unerasable-characters-iii-master
• hmtltoprint ○ coverImg ▪ comverImg1.jpg ○ font ▪ GenRyuMinTW-M-01.ttf ○ output.html ○ style.css • .gitlab-ci.yml • LICENSE • README.md	• code ○ assets ▪ HanyiSentyTang.ttf ○ Libraries ▪ p5.js ○ scrape ▪ weiboscope_scrape.py ○ index.html ○ silence.js ○ sketch.js • images ○ installation1.png ○ installation2.png • LICENSE • README.md • unersablecharactersII.gif	• DataProcessing ○ .gitkeep ○ cleaning_dividefile.py • visual ○ .gitkeep ○ unerasablecharacters3.png ○ unerasablecharacters3c.png ○ unnerasablecharactersIII.gif • WebPage ○ assets ▪ .gitkeep ▪ favicon.png ○ .gitkeep ○ index.html ○ style.css • README.md

Table 6: Unerasable Character Series analysed

As the tables depicted above clearly show, the file formats used in each individual artwork are spread across the board.

Unerasable Character Series, the newest and most modern of the three depicted artworks has the highest likelihood of successful, long-term preservation of its content and functionality.

From the 30 files in total, 16 fulfil the afore mentioned criteria for high probability for long-term preservation (A) Mortimore and Johns had defined—namely the .yml file, the two .py files, the license files, the .md files, .js files, and .pngs—these files do not need to be converted to another format.

Eight files with the extensions of .**gif**, .html, .css and .**jpg** can be sorted into file formats with medium probability for long-term preservation (B) – this makes them preferred over file formats coloured in red, but less desired than those coloured in green. The raster image picture files (.gif and .jpg) should be converted and preserved as either .png, uncompressed .tiff or lossless .jpeg200 files. The easiest solution for the three .html and two .css files would be to simply save them as plain text format files – either UTF-8 or .xml.

Only two files are assigned low probability for long-term preservation (C), namely the two .ttf files. – These should be converted to .svg files, a vector graphic file format with a higher preservation probability.

The two remaining .gitkeep files "[…] tell[s] github to do the opposite of its default behaviour, which is to ignore empty folders" (jlollis, 2024). Since they do not have content, they cannot and need not be preserved in any special way.

Two screenshots (Figure 27 and Figure 28) of the html code from *Unerasable Character Series I* can be seen here and accessed directly under this link[6] (Soon, 2022a).

Figure 27: Unerasable Characters I

[6] https://gitlab.com/siusoon/unerasable-characters-i/-/raw/main/htmltoprint/output.html?ref_type=heads

Figure 28: Unerasable Characters I

Figure 29, which we can see below, is a screenshot of the p5.js code from *Unerasable Character Series* II (accessible here[7]),

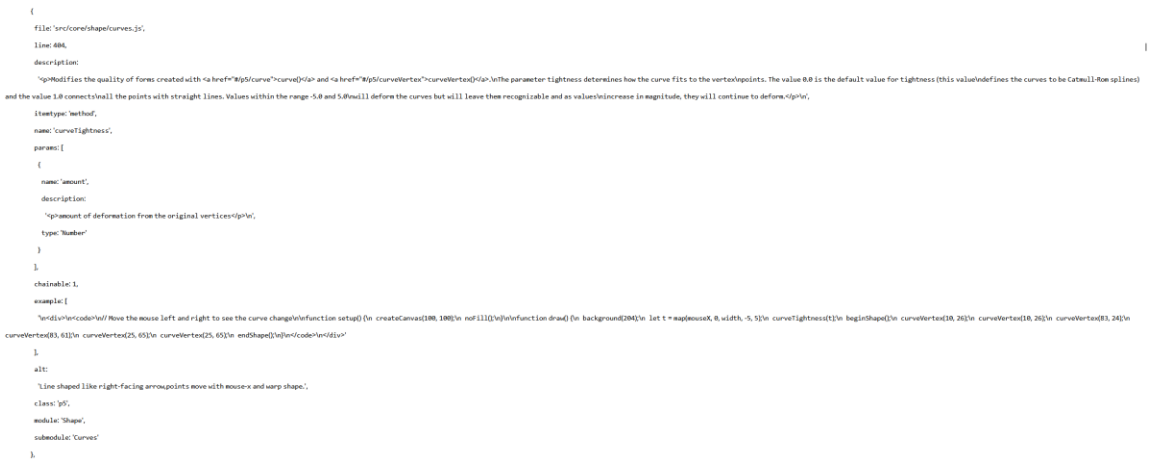

Figure 29: Unerasable Characters II

[7] https://raw.githubusercontent.com/siusoon/UnerasableCharactersII/refs/heads/master/code/libraries/p5.js

and the second picture, Figure 30, is a screenshot of the Python3 weiboscope scraper[8](Soon, 2020a, 2024).

```
  session = session or requests.Session()
  retry = Retry(
    total=retries,
    read=retries,
    connect=retries,
    backoff_factor=backoff_factor,
    status_forcelist=status_forcelist,
  )
  adapter = HTTPAdapter(max_retries=retry)
  session.mount('http://', adapter)
  session.mount('https://', adapter)
  return session


def processData( dataresponse, link ):
  soup2 = BeautifulSoup(dataresponse.content, 'html.parser')
  data = soup2.find('p')
  #extract Censored At:
  dataCensored = re.findall(r'<p><b>Ce.*?<p>', str(data))  #extract the field name
  dataCensored = re.sub(r'<[A-Za-z\/][^>]*>', '', str(dataCensored)) #remove html tags
  dataCensored = re.sub(r'Censored+\s+At:+\s', '', str(dataCensored)) #remove the field name
  dataCensored = re.sub(r'(\[\'|\'])','', str(dataCensored))  #remove '[]'
  #date_time_obj = datetime.datetime.strptime(dataCensored, '%Y-%m-%d %H:%M:%S.%f')
  if dataCensored !="":
    dataCensored = dataCensored[0:19]
    date_time_obj = datetime.datetime.strptime(dataCensored, '%Y-%m-%d %H:%M:%S')
    #compare time withtin 24 hours (LOG_timestamp - now and the data censored )
    current_time = LOG_timestamp.strftime('%Y %d %m %H %M %S')
    current_time = datetime.datetime.strptime(current_time,'%Y %d %m %H %M %S')
    censored_time = date_time_obj.strftime('%Y %d %m %H %M %S')
    censored_time = datetime.datetime.strptime(censored_time,'%Y %d %m %H %M %S')
    difference = current_time - censored_time
    minDiff = difference.total_seconds() / 60
    if minDiff < 1440:   #24 hours in the form of minutes
```

Figure 30: Unerasable Characters II

[8]https://raw.githubusercontent.com/siusoon/UnerasableCharactersII/refs/heads/master/code/scrape/weiboscope_scrape.py

This first screenshot, Figure 31, belonging to *Unerasable Character Series* III, is responsible for the cleaning and processing of the Covid-19 data collected by Weiboscope (fully accessible here[9] (Soon, 2021a)),

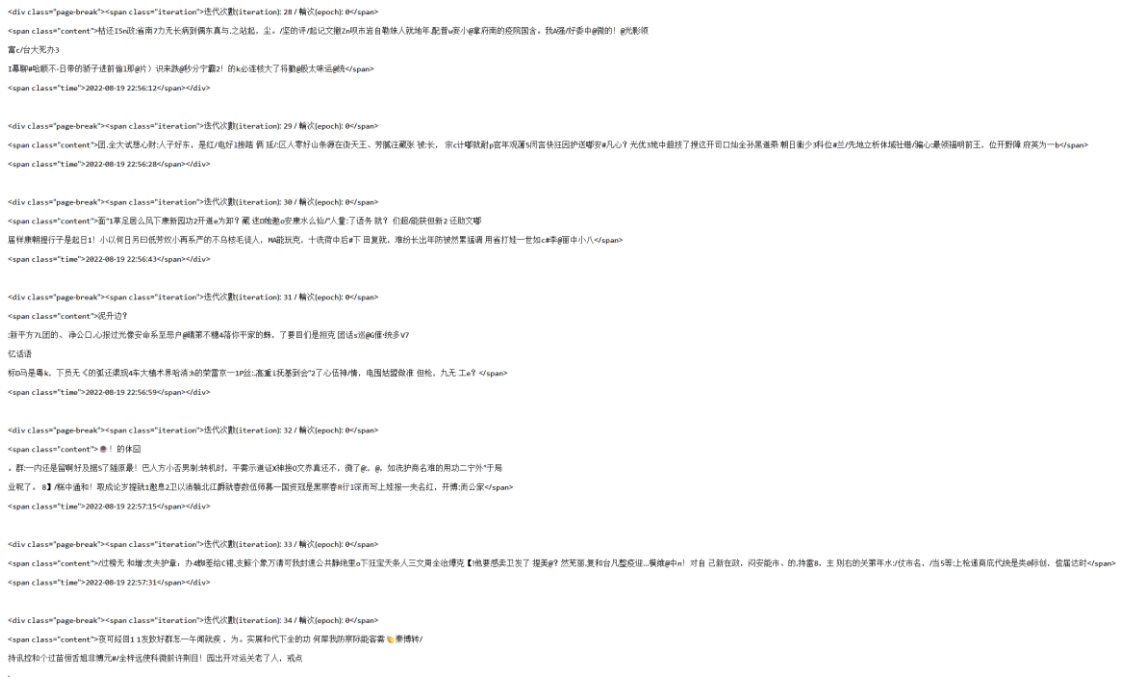

Figure 31: Unerasable Characters III

and this wall of html for constructing the webpage[10](Soon, 2021c):

Figure 32: Unerasable Characters III

[9] https://gitlab.com/siusoon/unerasable-characters-iii/
/raw/master/DataProcessing/cleaning_dividefile.py?ref_type=heads

[10] https://gitlab.com/siusoon/unerasable-characters-iii/-/raw/master/WebPage/index.html?ref_type=heads

5.5 Concluding Remarks

As the three case studies depicted above clearly show, nearly palpable three-way tension exists between *TwiLight*, *Unerasable Character Series*, and *VR Tour through the Doomsday Clock*. *TwiLight* and *Unerasable Character Series* are assembled from the same base material—source code—while *VR Tour through the Doomsday Clock* is "only" moving pictures.

In the case of *TwiLight*, the goal was to demonstrate a successful software emulation and present the source code employed in the original artwork. This endeavour did not succeed, caused by unsuccessfully preserved project files and the legacy code. It might have been easier to emulate it on an original, intended Atari machine, but this should not have to be the case.

The *Unerasable Character Series* could be described as the opposite of *TwiLight*: available source code with good documentation running on a modern, contemporary system. However, this current state should not fool us: only because the artwork series is stored on Github and is currently being supported by modern systems, no one can know what the general software environment will look like in 30 years. To not let it come so far as with *TwiLight*, the best documentation possible should be ensured – this includes videos as additional documentation (like it is the case for *VR Tour through the Doomsday Clock*), which has already been mentioned numerous times in Chapter 3, in addition to the available source code.

VR Tour through the Doomsday Clock was an interesting contrast to the other two case studies. Again, like in the case of *TwiLight*, no source code was available, only pictures and videos as documentation. The use of as many pictures as possible strived for an approximation of the original moving, immersive artwork.

6 Conclusion

The central focus of this thesis was to investigate how software-based artworks could be preserved and to illuminate the different, already existing methods.
This was undertaken from various directions: theoretical methodology was established by embedding software-based art in the bigger context of new media art and by thoroughly reviewing appropriate, contemporary literature to form a solid basis of understanding regarding preservation in general, contemplating the different existing types of software and source code and revising various, present strategies for the preservation of artworks while keeping their individual needs and requirements in mind. This theoretical framework was followed by a digression into the institutional field of Ars Electronica as a whole, the museum which planted the intellectual seed for this thesis.
The most extensive part consisted of the practical approach: three different case studies were examined in detail and looked at from every possible preservation angle. These case studies hooked onto the theoretical methods formulated earlier by directly applying the discourse established in a practical manner.
The main goal was to shed light on the endangered species of software-based artworks as a whole and to examine the current state of theoretical discussions in combination with actual preservation methods by employing a selection of them in a practical manner.
The limitations faced were few but nonetheless hindered in their own way, one being a mixture of software and hardware limitations, the other being limited to software only. The first software stumbling block was encountered during the case study of *TwiLight*, where the artwork could not be successfully emulated. After thorough research and contacting the co-executive director of the responsible institution, emulation was ultimately concluded as unsuccessful, with the result that *TwiLight* is not in a preserved state – contrary to what it said on the internet, which was further underlined by the successful emulation of a different Atari software. The hardware aspect of this consisted of not being in the possession of a legacy Atari machine, which would have made the preservation likely possible.
The second limitation was not being able to access the actual project files for a selected artwork, leaving only the documentation files open for analysis. This was done to the best of my abilities and resulted in a satisfactory outcome.

This thesis gives a future outlook on the state of preservation of software-based art by picking apart three individual artworks and asking the question of what an artwork really is – is source code the artwork? Or is the manifestation of source code the artwork? How can the unstoppable machine of software-based art decay be stopped? Furthermore, keeping source code in mind, space has been created for an alternative conservation method that has not yet been used and recognised in the art world as much as it deserves.

These are all necessary questions for a successful and enduring preservation methodology to be answered in the future.

It can be concluded that the quintessence of software-based art can be defined rather concisely: the relation between art, software, and hardware needs to be determined anew every time a software-based artwork is first encountered. Questions about preservation standards need to be thought out and negotiated to the utmost degree to guarantee the best preservation strategy possible.

7 Bibliography

ADA. (2018). *VR Tour Through the Doomsday Clock*. ADA | Archive of Digital Art. https://digitalartarchive.at/database/work/4116/

Ali, M. R. (2005). Why teach reverse engineering? *SIGSOFT Softw. Eng. Notes*, *30*(4), 1–4. https://doi.org/10.1145/1082983.1083004

Alvar Freude | republica. (n.d.). https://re-publica.com/de/user/5936

Andersen, C. U. (2023). *SHAPE - Digital Activism*. https://darc.au.dk/projects/shape-digital-activism

Application Systems Heidelberg Software GmbH. (2025). *Application Systems Heidelberg | Homepage*. https://www.application-systems.de/

Aris, S., Aeini, B., & Nosrati, S. (2023). A Digital Aesthetics? Artificial Intelligence and the Future of the Art. *Journal of Cyberspace Studies*, *7*(2), 219–236. https://doi.org/10.22059/jcss.2023.366256.1097

Ars Electronica Linz GmbH & Co KG. (n.d.-a). *Ars Electronica Archive Relaunch*. Ars Electronica Archive. Retrieved 7 February 2025, from https://ars.electronica.art/archive/en/ars-electronica-archive-relaunch/

Ars Electronica Linz GmbH & Co KG. (n.d.-b). *History*. About Ars Electronica. Retrieved 7 February 2025, from https://ars.electronica.art/about/de/history/

Ars Electronica Linz GmbH & Co KG. (2010a, June 15). About Prix Ars Electronica. *Prix Ars Electronica*. https://ars.electronica.art/prix/en/about/

Ars Electronica Linz GmbH & Co KG. (2010b, August 19). *ARS Electronica | Center*. https://web.archive.org/web/20100819094135/http://www.aec.at/center_about_en.php

Ars Electronica Linz GmbH & Co KG. (2022a). *Ars Electronica Archiv*. https://archive.aec.at/prix/

Ars Electronica Linz GmbH & Co KG. (2022b, December 15). Prix Ars Electronica. *Prix Ars Electronica*. https://ars.electronica.art/prix/en/

Ars Electronica Linz GmbH & Co KG. (2022c, December 22). About Ars Electronica Festival. *Ars Electronica Festival*. https://ars.electronica.art/festival/en/about/

Ars Electronica Linz GmbH & Co KG. (2022d, December 22). Festival Archive. *Ars Electronica Festival*. https://ars.electronica.art/festival/en/archive/

Ars Electronica Linz GmbH & Co KG. (2023a). *Ars Electronica Archive Relaunch—Project Report*.

Ars Electronica Linz GmbH & Co KG. (2023b). *Unerasable Characters Series*. https://archive.aec.at/prix/278850/

Ars Electronica Linz GmbH & Co KG. (2023c, June 19). It Could Be You. *S+T+ARTS Prize*. https://ars.electronica.art/starts-prize/en/it-could-be-you/

Ars Electronica Linz GmbH & Co KG. (2023d, October 30). Ars Electronica Futurelab. *Ars Electronica Futurelab*. https://ars.electronica.art/futurelab/en/

Ars Electronica Linz GmbH & Co KG. (2024a). *About Ars Electronica Archive*. Ars Electronica Archive. https://ars.electronica.art/archive/en/about/

Ars Electronica Linz GmbH & Co KG. (2024b). *Ars Electronica Archive*. Ars Electronica Archive. https://ars.electronica.art/archive/en/

(art)n. (2022). *About | artn*. Artnlab. https://www.artn.com/about

Bitberry Software Aps. (2010a, 2025). *GIF File: How to open GIF file (and what it is)*. File.Org. https://file.org/extension/gif

Bitberry Software Aps. (2010b, 2024). *H File: How to open H file (and what it is)*. File.Org. https://file.org/extension/h

Bitberry Software Aps. (2010c, 2024). *PRG File: How to open PRG file (and what it is)*. File.Org. https://file.org/extension/prg

Blank, M. (2008). *Impressum—MBs Weblog* -. https://www.megablank.de/impressum/

Bolnick et al. (2023). *A guide to reproducible archiving of data and code*. https://jevbio.net/information/a-guide-to-reproducible-archiving-of-data-and-code/

British Art Network. (n.d.). *Winnie Soon*. Winnie Soon - British Art Network. Retrieved 3 March 2025, from https://britishartnetwork.org.uk/membership/members/5912/

Campbell, S., & Hellar, M. (2019, May). *Virtual Reality Artwork Acquisition Template*. https://docs.google.com/document/d/1VGHHvxMRQc_OQYjHFW1mTMRa2vL9MGsjiNq1gF4j6l8/edit?usp=embed_facebook

Castagné, M. (2013). Consider the Source: The Value of Source Code to Digital Preservation Strategies. *School of Information Student Research Journal*, *2*(2). https://doi.org/10.31979/2575-2499.020205

centerforartlaw. (2024, April 16). *Artificial Intelligence versus/& Human Artists: AI as a Creative Collaborator in Art - Center for Art Law*. https://itsartlaw.org/2024/04/16/artificial-intelligence-versus-human-artists-ai-as-a-creative-collaborator-in-art/

Cheng, H. (2024). *It Could Be You—HsienYu Cheng ｜ 鄭先喻*. It Could Be You. https://chenghsienyu.com/it-could-be-you/

Coko. (2019). *What is Paged.js?* Paged.Js. https://pagedjs.org/documentation/1-the-big-picture/site.url

Delirium Arts. (1991). *TwiLight* [ATARI TOS].

Dell, P. (2009, 2024). *WUDSN - 8-bits are enough—Coding, demos and fun with Atari computers*. https://www.wudsn.com/index.php?platform=vcs&file=examples/complexscene2&rCH=2&start=35

Di Cosmo, R., & Zacchiroli, S. (2018). Software Heritage: Why and How We Collect, Preserve and Share All the Software Source Code. *ICSE-SEIS*, 11.

Digital Preservation Coalition. (2024a). *Exhibition Content—Digital Preservation Coalition*. https://www.dpconline.org/digipres/champion-digital-preservation/bit-list/critically-endangered/bitlist-exhibition-content

Digital Preservation Coalition. (2024b). *Manuals, Documentation, and Associated Materials—Digital Preservation Coalition*. https://www.dpconline.org/digipres/champion-digital-preservation/bit-list/critically-endangered/bitlist-manuals-docs-associated-materials

Digital Preservation Coalition. (2024c). *Recently Commissioned or Completed Media Art—Digital Preservation Coalition*. https://www.dpconline.org/digipres/champion-digital-preservation/bit-list/vulnerable/bitlist-media-art

Digital Preservation Coalition. (2024d). *The Global 'Bit List' of Endangered Digital Species—Digital Preservation Coalition*. https://www.dpconline.org/digipres/champion-digital-preservation/bit-list

Digital Preservation Coalition. (2025a). *Glossary—Digital Preservation Handbook*. https://www.dpconline.org/handbook/glossary

Digital Preservation Coalition. (2025b). *Why does digital preservation matter? - Digital Preservation Coalition*. What Is Digital Preservation? https://www.dpconline.org/digipres/what-is-digipres

drx.a-blast. (2008). *drx: TwiLight*. http://drx.a-blast.org/~drx/projects/twilight/index.de.html

drx.a-blast. (2012). *TwiLight*. http://static-files.rhizome.org/Q2862/1x-upon.com/~despens/twilight/

Electronic Arts Intermix. (n.d.-a). *Electronic Arts Intermix: Behavior*. EAI : Resource Guide. Retrieved 10 February 2025, from https://www.eai.org/webpages/1104

Electronic Arts Intermix. (n.d.-b). *Electronic Arts Intermix: Documentation*. EAI : Resource Guide. Retrieved 10 February 2025, from https://www.eai.org/webpages/1103

Electronic Arts Intermix. (n.d.-c). *Electronic Arts Intermix: Inspection*. EAI : Resource Guide. Retrieved 10 February 2025, from https://www.eai.org/webpages/1107

Electronic Arts Intermix. (n.d.-d). *Electronic Arts Intermix: Metadata*. EAI : Resource Guide. Retrieved 11 February 2025, from https://www.eai.org/webpages/1105

Electronic Arts Intermix. (n.d.-e). *Electronic Arts Intermix: Preservation Activities*. EAI : Resource Guide. Retrieved 11 February 2025, from https://www.eai.org/webpages/1109

Electronic Arts Intermix. (n.d.-f). *Electronic Arts Intermix: Preservation Strategies—Emulation*. EAI: Resource Guide. Retrieved 14 February 2025, from https://www.eai.org/webpages/1110

Electronic Arts Intermix. (n.d.-g). *Electronic Arts Intermix: Preservation Strategies—Migration*. EAI : Resource Guide. Retrieved 8 February 2025, from https://www.eai.org/webpages/1110

E-Maculation. (2025). *E-Maculation Home: All About Macintosh Emulation*. https://www.emaculation.com/home.html

Engel, D., & Phillips, J. (2018). *Introducing 'Code Resituation': Applying the Concept of Minimal Intervention to the Conservation Treatment of Software-based Art – Electronic Media Review*. https://resources.culturalheritage.org/emg-review/volume-5-2017-2018/engel-2/

Engel, D., & Phillips, J. (2023). *Conservation of time-based media art*. Routledge,. https://doi-org.uaccess.univie.ac.at/10.4324/9781003034865

Engel, D., & Wharton, G. (2014). Reading between the lines: Source code documentation as a conservation strategy for software-based art. *Studies in Conservation*, *59*(6), 404–415. https://doi.org/10.1179/2047058413Y.0000000115

Engel, D., & Wharton, G. (2015). SOURCE CODE ANALYSIS AS TECHNICAL ART HISTORY. *Journal of the American Institute for Conservation*, *54*(2), 91–101. https://doi.org/10.1179/1945233015Y.0000000004

Ensom et al. (2022). *Software-based Art Preservation*. Tate. https://www.tate.org.uk/about-us/projects/software-based-art-preservation

Ensom, T. (2021). *Disk Imaging Guide*. Tate Modern. https://www.tate.org.uk/documents/3/sbapp_disk_imaging_guide_01_00.pdf

Espenschied, D. (2012). *TwiLight—Rhizome Artbase*. https://artbase.rhizome.org/wiki/Q2861

Falcão, P. (2010). *Developing a Risk Assessment Tool for the conservation of software-based artworks.* https://www.academia.edu/6660777/Developing_a_Risk_Assessment_Tool_for_the_conservation_of_software_based_artworks

Falcão, P. (2011). *Risk Assessment as a Tool in the Conservation of Software-Based Artworks – Electronic Media Review*. https://resources.culturalheritage.org/emg-review/volume-two-2011-2012/falcao/

Falcão, P. (2019). *Preservation of Software–based Art at Tate.* https://ubdata.univie.ac.at/AC16094323

Falcão, P. (2024). *Preserving Digital Art*. https://doi.org/10.7207/twgn24-02

Falcão, P., Dekker, A., & Laurenson, P. (2016). *An Exploration of Significance, Dependency, and Virtualization in the Conservation of Software-based Artworks*

– *Electronic Media Review*. https://resources.culturalheritage.org/emg-review/volume-4-2015-2016/falcao/

File-Extensions.org. (2024a). *1ST file extension. How to open and convert files with 1ST file suffix*. https://www.file-extensions.org/1st-file-extension-readme-first-text-document

File-Extensions.org. (2024b). *MD file extension. How to open and convert files with MD file suffix*. https://www.file-extensions.org/md-file-extension

File-Extensions.org. (2024c). *OVL file extension. How to open and convert files with OVL file suffix*. https://www.file-extensions.org/ovl-file-extension

File-Extensions.org. (2024d). *PRJ file extension. How to open and convert files with PRJ file suffix*. https://www.file-extensions.org/prj-file-extension

File-Extensions.org. (2024e). *PY file extension. How to open and convert files with PY file suffix*. https://www.file-extensions.org/py-file-extension

File-Extensions.org. (2024f). *RSH file extension. How to open and convert files with RSH file suffix*. https://www.file-extensions.org/rsh-file-extension

File-Extensions.org. (2024g). *TTF file extension. How to open and convert files with TTF file suffix*. https://www.file-extensions.org/ttf-file-extension

File-Extensions.org. (2024h). *YML file extension. How to open and convert files with YML file suffix*. https://www.file-extensions.org/yml-file-extension

File-Extensions.org. (2025a). *CSS file extension. How to open and convert files with CSS file suffix*. https://www.file-extensions.org/css-file-extension

File-Extensions.org. (2025b). *HTML file extension. How to open and convert files with HTML file suffix*. https://www.file-extensions.org/html-file-extension

File-Extensions.org. (2025c). *JPG file extension. How to open and convert files with JPG file suffix*. https://www.file-extensions.org/jpg-file-extension

File-Extensions.org. (2025d). *MP4 file extension. How to open and convert files with MP4 file suffix*. https://www.file-extensions.org/mp4-file-extension

File-Extensions.org. (2025e). *PNG file extension. How to open and convert files with PNG file suffix*. https://www.file-extensions.org/png-file-extension

File-Extensions.org. (2025f). *TXT file extension. How to open and convert files with TXT file suffix*. https://www.file-extensions.org/txt-file-extension

File-Extensions.org. (2025g). *WEBP file extension. How to open and convert files with WEBP file suffix*. https://www.file-extensions.org/webp-file-extension

FILExt. (2024). *ACC File Extension - What is it? How to open an ACC file?* https://filext.com/file-extension/ACC

Freude, A. (2002). *Alvar C.H. Freude: Media Artist, Coder, Designer, Internet Developer, ...* https://alvar.a-blast.org/index.en.html

Friel, A. (2020). *It's not GPU pass-through, it's paravirtualization. | Hacker News*. https://news.ycombinator.com/item?id=23568006

Geater, J. (2023). *SMP File Extension: What Is It & How To Open It?* https://www.solvusoft.com/en/file-extensions/file-extension-smp/

GeeksforGeeks. (2024). *Code Migration in Distributed System*. GeeksforGeeks. https://www.geeksforgeeks.org/code-migration-in-distributed-system/

Gholizadeh, A., Sandor, E., & art(n). (2018). *It is Two Minutes to Midnight—An Exhibition of PHSCologram Sculptures and Virtual Reality Experiences*. 25.

GNU. (n.d.). *GNU General Public License v2.0—GNU Project—Free Software Foundation*. Retrieved 18 April 2024, from https://www.gnu.org/licenses/old-licenses/gpl-2.0.html

Grau, O. (2016). *New Media Art*. https://www.oxfordbibliographies.com/display/document/obo-9780199920105/obo-9780199920105-0082.xml

Grover, T. N. (2008). *Dream of the Techno-Shaman*. https://api.semanticscholar.org/CorpusID:191864485

Hashemi-Pour, C. (2024). *What Is Software? | Definition from TechTarget*. Search App Architecture. https://www.techtarget.com/searchapparchitecture/definition/software

Hashemi-Pour et al. (2023). *What is Docker and How Does It Work?* Search IT Operations. https://www.techtarget.com/searchitoperations/definition/Docker

Hirsch, A. J. (2019). *Creating the future: A brief history of Ars Electronica 1979-2019* (G. Stocker & D. Schwarzmair, Eds.). Ars Electronica, Berlin. Hatje Cantz Verlag.

HOL TEAM. (1990). *Mega Twins*. Hall of Light. https://amiga.abime.net/games/view/mega-twins

ICCROM. (2016). *Guide to Risk Management | ICCROM*. https://www.iccrom.org/publication/guide-risk-management

ISEA. (2022). Artwork / Liquid Views. *Artwork / Liquid Views*. https://isea2022.isea-international.org/event/artwork-liquid-views/

Jarczyk, A., Obermann, A., Engel, D., & Haage, K. (2017). *Interdisciplinary Discussions about the Conservation of Software-Based Art Community of Practice on Software-Based Art*. https://api.semanticscholar.org/CorpusID:27251567

jlollis. (2024). *.gitkeep—Push your entire folder structure to GitHub, including empty folders*. Gist. https://gist.github.com/jlollis/54d3b3a7ed12776ac9f0d4bc13e63ed9

Johns, E. (2025). *LibGuides: eCommons: Cornell's Digital Repository: Recommended file formats*. https://guides.library.cornell.edu/ecommons/formats

King-wa Fu, & Yuner Zhu. (2020). *COVID-19 related Weibo Data from 'Did the world overlook the media's early warning of COVID-19?'* [Dataset]. HKU Data Repository. https://doi.org/10.6084/m9.figshare.12199038.v1

Kunstuniversität Linz. (2012). *Bestoff12: Ausgewählte Studierendenarbeiten der Kunstuniversität Linz*. http://www.bestoff12.ufg.ac.at/index.php?menu=3&depth=3&arb_id=1706

Langsdorf, S. M. (2015). *Martyl*. MARTYL. http://www.martyl.com/about.html

Laurenson, P. (2006). *Authenticity, Change and Loss in the Conservation of Time-Based Media Installations – Tate Papers*. Tate. https://www.tate.org.uk/research/tate-papers/06/authenticity-change-and-loss-conservation-of-time-based-media-installations

Laurenson, P. (2014a). *Old Media, New Media? Significant Difference and the Conservation of Software-Based Art* (B. Graham, Ed.; 1st ed., p. 96). https://doi.org/10.4324/9781315597898-4

Laurenson, P. (2014b). Old Media, New Media? Significant Difference and the Conservation of Software-Based Art. In *New Collecting: Exhibiting and Audiences after New Media Art*. Routledge.

Liakopoulos, K. (2023, July 19). *Intersection of AI and VR/AR: Exploring the Synergy*. Medium. https://medium.com/@wiz-wizdomgr/intersection-of-ai-and-vr-ar-exploring-the-synergy-89dc17820aad

Lutkevich, B., & Wigmore, I. (2022). *What is an executable file (EXE file)?* WhatIs. https://www.techtarget.com/whatis/definition/executable-file-exe-file

Mathias. (2008). *Atari Screensaver for Firebee—Atari-Forum*. https://atari-forum.com/viewtopic.php?p=246091&sid=03029e65db39549b8e4a0cced0fd54c2#p246091

Mee, M. (2024). *Emulators—Atari Wiki*. https://www.atari-wiki.com/index.php?title=Emulators

Mell, E. (2023). *What is Docker Engine?* Search IT Operations. https://www.techtarget.com/searchitoperations/definition/Docker-Engine

Mission. (n.d.). About Ars Electronica. Retrieved 28 November 2024, from https://ars.electronica.art/about/en/

MoMa. (2023). *Matters in Media Art | MoMA*. The Museum of Modern Art. https://www.moma.org/research/conservation/matters-in-media-art

Mortimore, J. (2015). *Library Guides: Data Management Services: Recommended File Formats for Long-Term Data Curation*. https://georgiasouthern.libguides.com/c.php?g=833713&p=5953146

Mürer, K., Bernard, E., & Ludwig, T. (2021). Archivierung Digitaler Kunst aus Museumssicht. *Informatik Spektrum*, *44*(1), 38–43. https://doi.org/10.1007/s00287-021-01342-2

Owens, T. (2014, March 24). *"Digital Culture is Mass Culture": An interview with Digital Conservator Dragan Espenschied | The Signal* [Webpage]. The Library of Congress. https://blogs.loc.gov/thesignal/2014/03/digital-culture-is-mass-culture-an-interview-with-digital-conservator-dragan-espenschied

p5.js Contributors et al. (2021). *p5.js—About*. https://p5js.org/about/

Rechert, K., Falcão, P., & Ensom, T. (2016). *Introduction to an emulation-based preservation strategy for software-based artworks*. Tate Modern.

Rhizome. (2016). *On your wrist is the universe—Rhizome Artbase*. On Your Wrist Is the Universe. https://artbase.rhizome.org/wiki/Q12174

Rhizome. (n.d.). *About Rhizome*. Rhizome. https://rhizome.org/

Röck, C. (2020, July 24). *Interview with Claudia Röck by Tom Ensom and Patrícia Falcão*.

Rossenova, L. (2020). *ArtBase Archive—Context and History: Discovery Phase and User Research 2017– 2019.* (p. 118). Rhizome. https://sites.rhizome.org/artbase-re-design/docs/1_Report_ARTBASE-HISTORY_2020.pdf

Russell, R., & Winkworth, K. (2009). *Significance 2.0: A guide to assessing the significance of collections* (2. ed). Collections Council of Australia.

Sandor, E., art(n), Kemp, C., Torres, D., Gholizadeh, A., & Fron, J. (2017). *Have a Nice Day II PHSCologram Sculpture* [Virtual Photograph/Digital PHSCologram Sculpture, Duratrans, Kodalith and Plexiglas]. https://www.artn.com/virtual-sculpture

Sandor, E., art(n), Martyl, Miller, K., Latrofa, P., & Fron, J. (2002). *Have a Nice Day* [Digital PHSCologram, Duratrans, Kodalith and Plexiglas]. https://www.visualizinginanewlight.com/haveaniceday

Sandor, E., art(n), Torres, D., & Gholizadeh, A. (2018). *VR Tour Through the Doomsday Clock* [VR Tour]. https://www.artn.com/it-is-two-minutes-to-midnight/

Serexhe, B. (n.d.). *Konservierung von Medienkunst | ZKM*. Retrieved 6 February 2025, from https://zkm.de/de/schwerpunkt/konservierung-von-medienkunst

Sheldon et al. (2024). *What is application containerization (app containerization)?* Search IT Operations. https://www.techtarget.com/searchitoperations/definition/application-containerization-app-containerization

Sheldon, R. (2024). *What is paravirtualization?* Search IT Operations. https://www.techtarget.com/searchitoperations/definition/paravirtualization

Soares, C., & Simão, E. (2020). Software-Based Media Art: From the Artistic Exhibition to the Conservation Models. In *Multidisciplinary Perspectives on*

New Media Art (pp. 47–63). IGI Global Scientific Publishing. https://doi.org/10.4018/978-1-7998-3669-8.ch003

Soon, W. (n.d.). *About | Winnie Soon | 孫詠怡*. Retrieved 3 March 2025, from https://siusoon.net/about

Soon, W. (2020a). *Libraries—P5.js—Unerasable Characters II* (Version 1.0 [Source code)) [Computer software]. https://raw.githubusercontent.com/siusoon/UnerasableCharactersII/refs/heads/master/code/libraries/p5.js

Soon, W. (2020b). *Unerasable Characters II / 刪不了的符號 II | Winnie Soon | 孫詠怡* . https://siusoon.net/projects/unerasablecharacters-ii

Soon, W. (2021a). *cleaning_dividefile—Unerasable Characters III* (Version 2.0 [Source code]) [Computer software]. https://gitlab.com/siusoon/unerasable-characters-iii/-/raw/master/DataProcessing/cleaning_dividefile.py?ref_type=heads

Soon, W. (2021b). *Unerasable Characters III / 刪不了的符號 III | Winnie Soon | 孫詠怡*. https://siusoon.net/projects/unerasablecharacters-iii

Soon, W. (2021c). *WebPage—Index—Unerasable Characters III* (Version 3.0 [Source code]) [Computer software]. https://gitlab.com/siusoon/unerasable-characters-iii/-/raw/master/DataProcessing/cleaning_dividefile.py?ref_type=heads

Soon, W. (2022a). *Htmltoprint—Output—Unerasable Characters I* (Version 2.0 [Source code]) [Computer software]. https://gitlab.com/siusoon/unerasable-characters-i/-/raw/main/htmltoprint/output.html?ref_type=heads

Soon, W. (2022b). *Unerasable Characters I / 刪不了的符號 I | Winnie Soon | 孫詠怡*. https://siusoon.net/projects/unerasablecharacters-i

Soon, W. (2023). *Unerasable Characters Series*. Unerasable Characters Series. https://archive.aec.at/prix/278850/

Soon, W. (2024). *weiboscope_scrape—Unerasable Characters II* (Version 11.0 [Source code]) [Computer software]. https://raw.githubusercontent.com/siusoon/UnerasableCharactersII/refs/heads/master/code/scrape/weiboscope_scrape.py

ST Computer 05. (1987). *ST-Computer :05/1987 ST-Ecke: HEADER-Files des Resource-Construction-Set*. https://www.stcarchiv.de/stc1987/05/st-ecke

Sumpter, G., & et al. (1993). *Atari ST Review 09*. http://archive.org/details/atari-st-review-009

superuser1931. (2009, October 27). *Answer to 'How to merge/combine files with extensions .001, .002, .003.. Etc?'* [Online post]. Super User. https://superuser.com/a/61535

Tate. (2021). *Time-based media*. Tate. https://www.tate.org.uk/art/art-terms/t/time-based-media

Tate. (2022). *Matters in Media Art*. Tate. https://www.tate.org.uk/about-us/projects/matters-media-art

The Solomon R. Guggenheim Foundation. (2020). *The Conserving Computer-Based Art Initiative*. The Guggenheim Museums and Foundation. https://www.guggenheim.org/conservation/the-conserving-computer-based-art-initiative

Tiefengraber, S. (2011a, 2012). *Hello, I am ... Our Winner: Stefan Tiefengraber, Your unerasable text.* Austrian Cultural Forum Washington Events Archive. https://www.archive.acfdc.org/hello-i-am/winner/stefan-tiefengraber/work-2

Tiefengraber, S. (2011b, 2012). *Your unerasable text*. A New Digital Deal. https://ars.electronica.art/newdigitaldeal/en/your-unerasable-text/

Time-based Media Conservation, Tate. (2020a). *Disk Imaging Report*. Tate Modern. https://www.tate.org.uk/documents/4/sbapp_imagingreport_template_01_00.pdf

Time-based Media Conservation, Tate. (2020b). *Emulation Report*. https://www.tate.org.uk/documents/6/sbapp_emulationreport_template_01_00.pdf

ual. (n.d.). *Winnie Soon | About | University of the Arts London staff research profiles*. Retrieved 3 March 2025, from https://researchers.arts.ac.uk/2466-winnie-soon

UCL. (2018, August 8). *Software sustainability, preservation and sharing*. Library Services. https://www.ucl.ac.uk/library/open-science-research-support/research-data-management/best-practices/how-guides/software

vmware. (n.d.). *What is a Hypervisor?* Retrieved 14 February 2025, from https://www.vmware.com/topics/hypervisor

Wainwright, L., Balsamo, A., & Malloy, J. (2018). *New Media Futures*. University of Illinois Press; JSTOR. https://doi.org/10.5406/j.ctv8j4d3

Waller, R. (1994). Conservation risk assessment: A strategy for managing resources for preventive conservation. *Studies in Conservation*, *39*(sup2), 12–16. https://doi.org/10.1179/sic.1994.39.Supplement-2.12

Zabolitzky, J. G. (2002). *Preserving Software: Why and How*. https://api.semanticscholar.org/CorpusID:7432433

ZKM Karlsruhe. (n.d.-a). *Labor für antiquierte Videosysteme | ZKM*. Retrieved 7 February 2025, from https://zkm.de/de/labor-fuer-antiquierte-videosysteme

ZKM Karlsruhe. (n.d.-b). *Restaurierung elektronischer und digitaler Kunst | ZKM*. Retrieved 6 February 2025, from https://zkm.de/de/restaurierung-elektronischer-und-digitaler-kunst

8 List of Tables

9 List of Figures