Applying conservation ethics to the examination and treatment of software- and computer-based art

Deena Engel & Joanna Phillips

To cite this article: Deena Engel & Joanna Phillips (2019) Applying conservation ethics to the examination and treatment of software- and computer-based art, Journal of the American Institute for Conservation, 58:3, 180-195, DOI: 10.1080/01971360.2019.1598124

To link to this article: https://doi.org/10.1080/01971360.2019.1598124

Published online: 08 Jun 2019.

JOURNAL OF THE AMERICAN INSTITUTE FOR CONSERVATION
2019, VOL. 58, NO. 3, 180–195
https://doi.org/10.1080/01971360.2019.1598124

Applying conservation ethics to the examination and treatment of software- and computer-based art

Deena Engel [a] and Joanna Phillips [b]

[a]Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, New York, NY, USA; [b]Solomon R. Guggenheim Museum, New York, NY, USA

ABSTRACT
As part of the Solomon R. Guggenheim Museum's ongoing initiative Conserving Computer-based Art (CCBA) and the collaborative research between the museum and the Department of Computer Science of New York University's Courant Institute for Mathematical Sciences, Guggenheim media art conservators and NYU computer scientists have collaborated since 2014 to examine, document, and treat software- and computer-based artworks from the Guggenheim collection. Based on this interdisciplinary work and guided by the objective to inform the care for computer-based art within the field of contemporary art conservation, the authors propose new examination, documentation, and treatment practices for software- and computer-based art that accommodate established conservation ethics and practice guidelines.

RÉSUMÉ
Dans le cadre de l'initiative en cours du Solomon R. Guggenheim Museum intitulée « Conserving Computer-based ART (CCBA) » et de la recherche collaborative entre le musée et le Département de sciences informatiques du Courant Institute for Mathematical Sciences de la New York University (NYU), les restaurateurs d'art médiatique du Guggenheim et les informaticiens de la NYU ont collaboré depuis 2014 pour examiner, documenter, et traiter des œuvres d'art numérique et d'art logiciel dans la collection du Guggenheim. Basées sur ce projet interdisciplinaire et guidées par l'objectif d'informer sur le soin des œuvres d'art numérique dans le domaine de la conservation - restauration de l'art contemporain, les auteures proposent de nouvelles méthodes d'examen, de documentation, et de traitement pour les œuvres d'art numérique qui respectent les normes d'éthique et de pratique en conservation. Traduit par Eric Henderson.

RESUMO
Como parte da iniciativa contínua do Conserving Computer-based Art - CCBA (Conservação de obras de arte digitais) do Museu Solomon R. Guggenheim e a pesquisa colaborativa entre o museu e o Departamento de Ciências da Computação do Instituto Courant para Ciências Matemáticas da Universidade de Nova York, conservadores de arte e mídia do Guggenheim e cientistas da computação da NYU colaboram desde 2014 para examinar, documentar e tratar obras de arte digitais baseadas em softwares e computadores da coleção Guggenheim. Com base nesse trabalho interdisciplinar e orientado pelo objetivo de informar o cuidado com a arte digital no campo da conservação da arte contemporânea, os autores propõem novas práticas de exame, documentação e tratamento da arte criada com uso de software e computadores que acomodam as diretrizes praticadas e a ética estabelecida na conservação. Traduzido por Millar Schisler.

RESUMEN
Como parte de la iniciativa en curso Conservación del arte creado en computadora (CCBA – siglas del nombre en inglés Conserving Computer-based Art) del Solomon R. Guggenheim Museum y la investigación colaborativa entre el museo y el Departamento de Ciencias de la Computación del Instituto Courant de Ciencias Matemáticas de la Universidad de Nueva York, los conservadores de arte de medios del Guggenheim y los científicos de informática de la NYU han colaborado desde 2014 para examinar, documentar y tratar obras de arte basadas en software y computadoras de la colección Guggenheim. Apoyados en este trabajo interdisciplinario y guiados por el objetivo de informar el cuidado del arte creado en computadora dentro del campo de la conservación del arte contemporáneo, los autores proponen nuevos exámenes, documentación y prácticas de tratamiento para el arte basado en software y computadora que se adapten a las pautas y ética de conservación establecidas. Traducción: Amparo Rueda.

ARTICLE HISTORY
Received 7 August 2018
Accepted 18 March 2019

KEYWORDS
Software-based art; computer-based art; conservation ethics; time-based media conservation; media conservation; source code analysis

CONTACT Joanna Phillips joannaphil@gmail.com

1. Introduction

Previous collaborations between media art conservators and computer scientists have identified source code analysis of software-based art as a tool for technical art history (Engel and Wharton 2015), and source code documentation as a conservation strategy (Engel and Wharton 2014). Building on these fundamental discoveries, the research featured in this paper is an initial attempt to apply established conservation ethics and practice guidelines to the examination and treatment of software- and computer-based artworks. The focus of this study are invasive treatment scenarios, in which functional and behavioral damages to an artwork cannot be restored without changing, amending or migrating the original source code.

Among the emerging community of conservation professionals who engage with software- and computer-based art, there has been no established consensus on the value and status of original source code. A recent collaborative report questions "whether it was useful to collect the source code along with a work, or whether it might in some cases give collectors a false sense of security, or take up too many resources, possibly without enough returns … " (Dekker and Falcao 2016, 6). Challenging the value of original source code for an artwork's collection and preservation is rooted in two decades of critical thinking and new ethical frameworks in the conservation of contemporary art, and time-based media conservation in particular. These new frameworks advise conservation professionals to consider the artist's intent in their decision-making (SBMK 1999) instead of focusing exclusively on the conservation of the original object. The preservation of an artwork's artist-intended behaviors may be privileged over the maintenance of its original, technical constituents (The Variable Media Approach 2003), and conservators' identification of the significant, "work-defining properties" of media components (Laurenson 2006) may result in the determination that original components are replaceable with new or other equipment or technologies (Laurenson 2004; Phillips 2012). With this perspective, prior explorations of the "significant properties" of software-based artworks have focused on their appearance, behaviors, experience, and other discernible properties (Laurenson 2014, 92–94), but have not yet attributed significance to the original source code itself.

The research in section 2.1 will demonstrate that, in addition to the work's discernible behaviors, significance can be found in the original source code, namely the artist's or programmer's choice of technologies, programming languages and coding styles. Similar to other areas of art technology, these choices of medium can be deliberate (artist-intended), or contingent. In both cases, the original source code may be integral to an artwork's identity, even if it is typically hidden from the audience and rarely part of the audiovisual or interactive experience.

But if the original source code is considered significant and worthy of conservation, it then becomes imperative to investigate how code-based artworks can be approached with "informed respect" (AIC 1994, II) as described by the American Institute for Conservation's *Code of Ethics* and *Guidelines for Practice*, even when a conservation treatment requires invasive source code intervention. The challenge to develop treatment practices that respect the originality of source code is magnified by the fact that conservators have to delegate source code examinations and treatments to trained specialists such as computer scientists and programmers, because the expertise to read, interpret, and write source code is commonly not included in specialized media conservation training. Between the fields of art conservation and computer science, however, different core principles govern the approach to source code intervention, including the choice of treatment methods, the handling of defunct original code, and the documentation of interventions. Therefore, the adherence to basic conservation principles cannot be guaranteed, if delegated programming work is not informed and supervised by conservation professionals.

This cross-disciplinary study, co-authored by a media art conservator and a computer scientist, looks at a selection of core principles in art conservation and investigates how they can be applied to guide examination and treatment practices of software- and computer-based art. The research was carried out in the course of the Solomon R. Guggenheim Museum's ongoing initiative, Conserving Computer-based Art (CCBA), and its research collaboration with the Department of Computer Science of New York University's Courant Institute for Mathematical Sciences. The CCBA initiative is charged with examining, documenting, archiving and treating the Guggenheim's collection of software- and computer-based artworks (Guggenheim Blogs 2016).

2. The application of conservation ethics to the care of software- and computer-based art

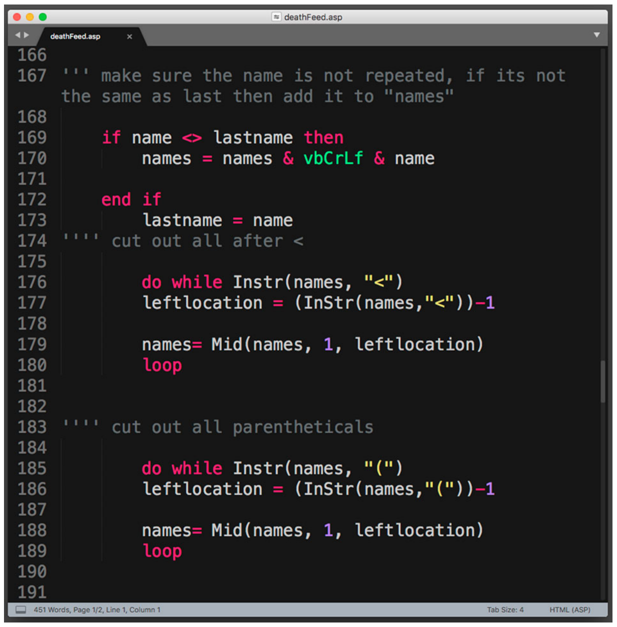
The adherence to national and international professional guidelines is integral to conservators' conduct, decision-making, and daily practice. As a reference framework for practice development in this investigation, the authors selected the *Code of Ethics, the Guidelines for Practice,* and the *Commentaries to the Guidelines for Practice* developed by the American Institute for Conservation of Historic and Artistic Works (AIC 1994). The authors selected these documents not only because the research was carried out within the North American conservation realm, but because they usefully provide guidance on a higher level without excluding artworks with non-physical or non-original components. In contrast, for example the guidelines of the European Confederation of Conservator-Restorers's Organisations (ECCO 2003) define artwork integrity more concretely as physical integrity and create more friction for media art conservation by categorically dismissing the removability of original elements (Phillips 2010; Weyer 2010).

It is outside of the scope of this paper to discuss the genesis of these documents (Appelbaum 2011), the discourse on limitations of popular conservation notions such as minimal intervention and reversibility (Villers 2004; Muñoz Viñas 2009), or ongoing considerations on the interpretation and application of ethics guidelines (Clavir 2002; Van de Vall 2009, 2015; Ashley-Smith 2017; Wharton 2018). The aim of this study is instead to take a widely accepted and established framework as a starting point for informed practice development in an emerging new conservation area that has otherwise seen little orientation along established conservation guidelines. In the following, six fundamental conservation principles are selected for application and closer examination: (1) conservator's required respect for the significant original (e.g. the artist's code); (2) the requirement to make informed treatment choices; (3) the requirement that conservation professionals must supervise and take responsibility for delegated conservation work; (4) a conservator's approach to loss compensation; (5) conservation's requirement to document the condition, examination and treatment of cultural property, and (6) the notion of preventive conservation.

2.1. Respect for the original

All actions of the conservation professional must be governed by an informed respect for the cultural property, its unique character and significance … (AIC 1994, II)

An artist's (or commissioned programmer's) individual use of technologies and programming languages can be compared to a painter's preference for certain painting materials and does not only determine the experience of the work, but also manifests a personal artistic practice. When analyzed, the original code of a software-based work offers unique insights into an artist's medium and technique, his or her individual programming preferences and style, the work's conceptual and technical genesis, and may also offer clues in regard to authorship, authenticity, and provenance. Additional important indications of an artist's intent may be



```
''' make sure the name is not repeated, if its not
the same as last then add it to "names"

    if name <> lastname then
        names = names & vbCrLf & name

    end if
        lastname = name
'''' cut out all after <

        do while Instr(names, "<")
        leftlocation = (InStr(names,"<"))-1

        names= Mid(names, 1, leftlocation)
        loop


'''' cut out all parentheticals

        do while Instr(names, "(")
        leftlocation = (InStr(names,"("))-1

        names= Mid(names, 1, leftlocation)
        loop

```

Figure 1. Screenshot of the original source code for Siebren Versteeg's *Untitled Film II* (2006), with the artist's source code annotations in gray. Courtesy of the Solomon R. Guggenheim Museum.

found in the artist's code annotations, human-readable comments in the code that may explain intended artwork behaviors or document dismissed drafts, but are ignored by the computer at runtime (Figure 1).

The style of coding as creative expression – similar to the method of paint application – significantly traces the artist's conceptual approaches to problem-solving and artistic path to achieving the intended (and unintended) behaviors and appearance of the final work. Similar to other areas of contemporary art conservation, these production processes remain significant, even if the artist outsourced the fabrication of the artwork, e.g. if the code was not authored by the artist herself, but rather by her collaborators or programmers.

The following three examples from the Guggenheim collection serve to illustrate some of these (often invisible) significances, which arguably require a conservator's identification and consideration in treatment planning. In the first example, *6th* ~~Light~~ (2007) by Paul Chan (b. 1973), the artist made a deliberate choice of technology, Adobe Flash, that he considers irreplaceable for his work (Guggenheim Collection Online n.d. Chan). In the second example, *Untitled Film II* (2006) by Siebren Versteeg (b. 1971), the artist preferred a programming language that is outdated but native to his practice and prevalent in his oeuvre. In the third example, the web artwork *Brandon* (1998–1999) by Shu Lea Cheang (b. 1954), the programming style reveals personal coding preferences within a multi-authored artwork structure (Guggenheim Collection Online n.d. Cheang).

2.1.1. Significance of the artist-selected technology

Paul Chan's Flash animation *6th* ~~Light~~ is projected onto the gallery floor and shows a window cross with shadows of small and large fragments floating past the window (Figure 2). On display, *6th* ~~Light~~ could be mistaken for a black-and-white video: the animated content is identical in every 14-minute loop. However, the computer-based piece is running an executable file, which was created in Adobe Flash. (An executable file, which is compiled from source code and is not human-readable, contains the program's instructions in a format that the computer can read directly.) The software application Adobe Flash, an earlier version of Adobe's Animate, allows an artist to animate images, shapes and sounds on a timeline and supports a programming language called ActionScript.

In the face of currently fading Flash support and concerns about the sustainability of *6th* ~~Light~~ (the executable file runs too fast on contemporary computers), conservators proposed to the artist that the Flash file could be exported and played back as a video file. This measure would recreate the identical visitor experience but make the work independent from its original software and hardware. For the artist, however, the Flash technology carries a conceptual significance and the proposed treatment was not an option:

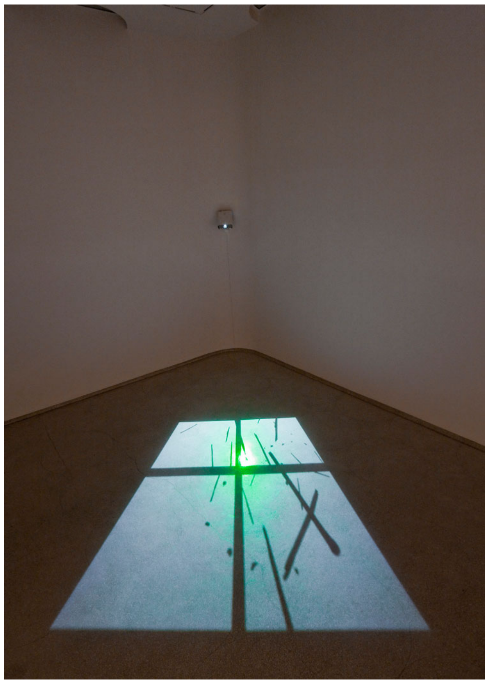


Figure 2. Paul Chan, *6th*, ~~Light~~ 2007 (2007.31). Projected Adobe Flash animation, silent, 14 min., dimensions variable. Courtesy of the Solomon R. Guggenheim Museum.

> I think it [the use of Flash] connects to my spartan mentality … I like Flash because of the control, and the precision, and the dirtiness of it. The roughness of the vector images. But I also love the fact that when I kick out that animation, it's like a 150 K … I don't want to ever imagine me making a work that would take up 10 TB. I want it to be as elegant and as compressed as possible. (Chan 2016)

The understanding of the conceptual significance of Flash technology for this artwork informs the treatment approach: here, emulation may be a more suitable strategy than migration. Emulation is a process of using software to imitate another program or environment, for example, to imitate an older operating system that is no longer in common use in order to run a program from that time. In the case of *6th* ~~*Light*~~, the original executable file would run within an emulated environment that simulates how it ran on the artist-provided legacy computer. Migration, which is the process of translating the artwork or any of its components from one technology or programming language to another, would be a more invasive treatment. This is particularly true for the case of *6th* ~~*Light*~~, as any migration, whether to video or to a different programming environment, would currently require the choice of a technology very different from the original Flash.

2.1.2. Significance of the artist-selected programming language

Siebren Versteeg's work *Untitled Film II* (2006) is an example of a work in which the specific programming language is significant to the artist. The artwork, creating the impression of start and end credits of a movie displayed on two screens, imports live data feeds from the Internet to display names of newborn infants on one screen and names of those recently deceased on the adjacent screen (Figure 3). The background images on the screen are randomly selected and the soundtrack is programmatically generated, both in real time. Versteeg wrote the work in the software application Macromedia Director using the programming language Lingo. Director was widely used in the 1990s to build on the Shockwave platform; the Macromedia Shockwave player plug-in was first released in 1995. Adobe purchased Director in 2005 and discontinued its sale and support in February 2017 (Miller 2017). When *Untitled Film II* was written in 2006, Lingo was already not in common use. However, the artist deliberately chose to write his work in Lingo:

> I still use Lingo. I still use Director … I mean I think in it now. When I start thinking about projects I think through that language and how that might be possible in that format … The authoring environment is so self-enclosed and simple, and there's no external

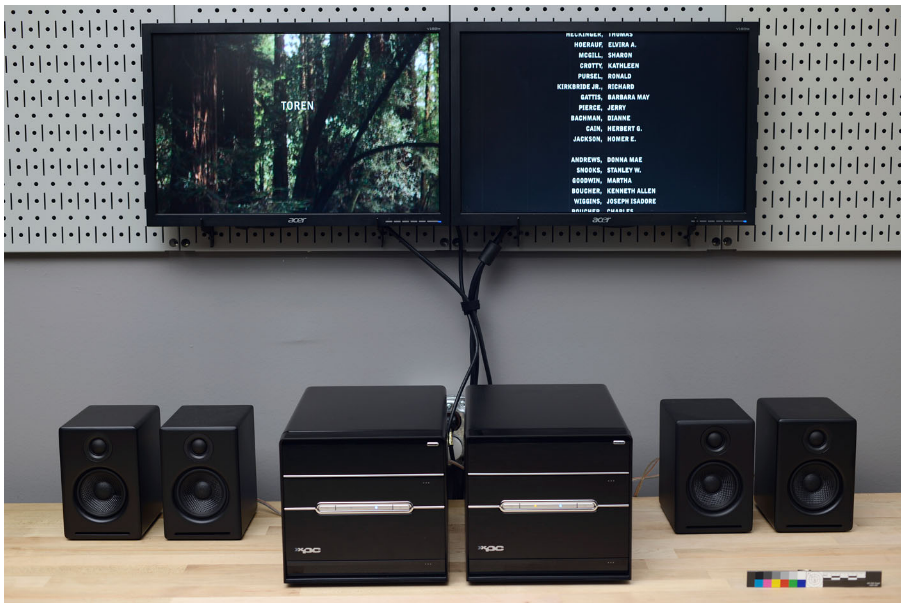


Figure 3. Lab set-up of Siebren Versteeg's *Untitled Film II*, 2006 (2006.75). Two internet-connected computers output to two LCD screens, real-time obituary listings, real-time birth announcements. Courtesy of the Solomon R. Guggenheim Museum.

> compilers needed and there's no external libraries needed … I like the solipsism of it … This is my medium. (Versteeg 2014)

Among programmers, it is common parlance to say that one has a "mother tongue" programming language; this resembles an artist's preference for a particular medium. A "real" Versteeg artwork is most likely programmed in Lingo. This lends significance to Lingo in the context of Versteeg's oeuvre in general and *Untitled Film II* in particular. In contrast, the same artwork's Active Server Pages (ASP) scripts on the server-side that scrape websites for infant and obituary names (Figure 4), have not been identified to carry any significance beyond delivering the scraping functionality. Therefore, current treatment testing considers emulation as a strategy to retain the original Lingo-based Director executable, but seeks

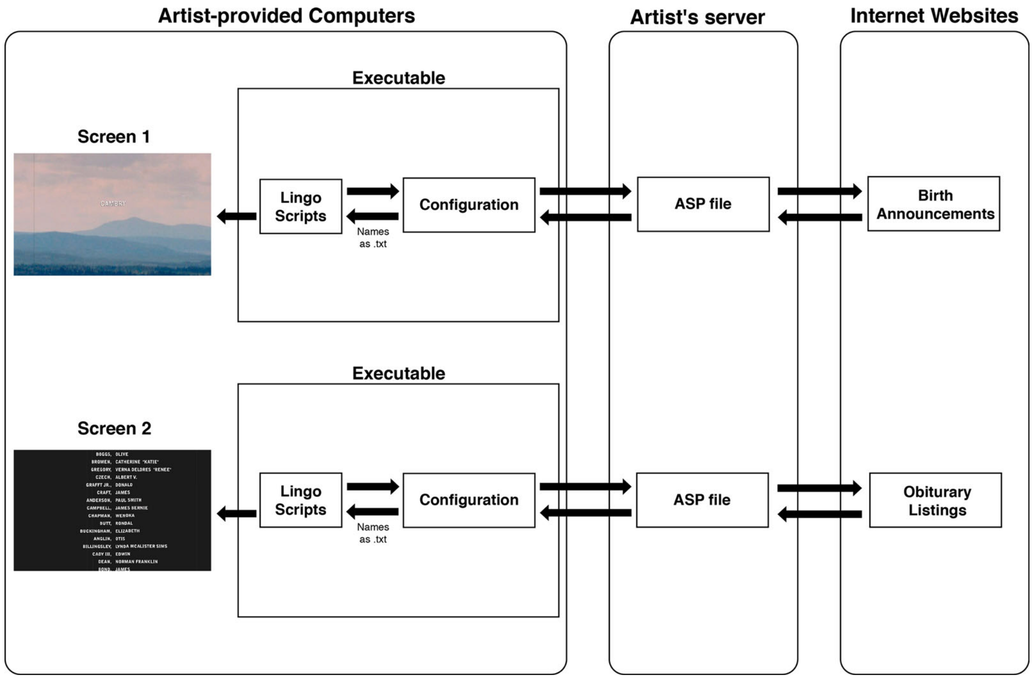
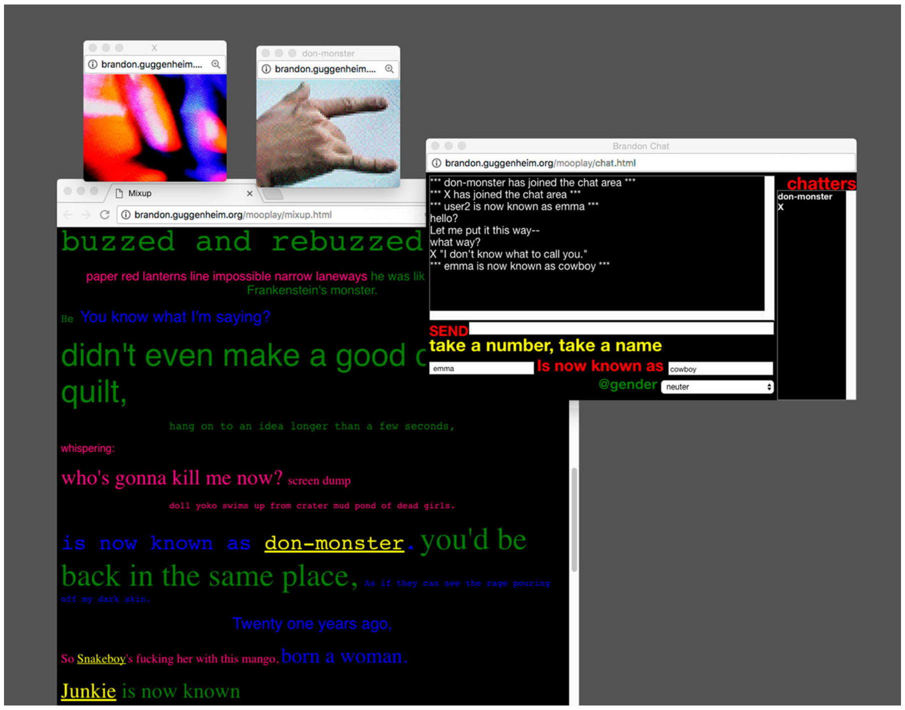


Figure 4. This functional diagram of Siebren Versteeg's *Untitled Film II* (2006) visualizes the anatomy of the artwork based on the findings of the source code analysis. Courtesy of the Solomon R. Guggenheim Museum.



*** don-monster has joined the chat area ***
*** X has joined the chat area ***
*** user2 is now known as emma ***
hello?
Let me put it this way--
what way?
X "I don't know what to call you."
*** emma is now known as cowboy ***

buzzed and rebuzzed

paper red lanterns line impossible narrow laneways he was lik
Frankenstein's monster.

He You know what I'm saying?

didn't even make a good c
quilt,

hang on to an idea longer than a few seconds,

whispering:

who's gonna kill me now? screen dump

doll yoko swims up from crater mud pond of dead girls.

is now known as don-monster. you'd be back in the same place, As if they can see the rage pouring off my dark skin.

Twenty one years ago,

So Snakeboy's fucking her with this mango, born a woman.

Junkie is now known

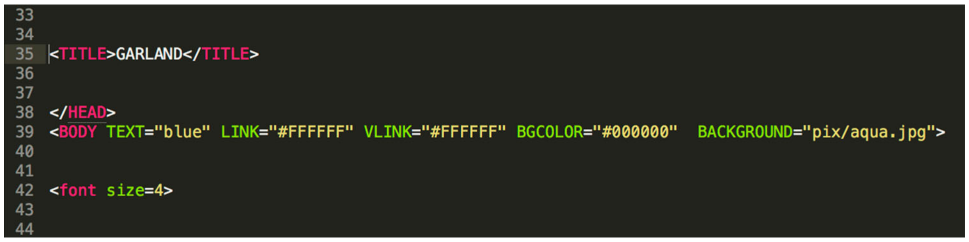
Figure 5. Screenshot of Shu Lea Cheang, *Brandon,* 1998–1999 (2005.44). Interactive networked code (HTML, Java, Javascript and server database). The work is publicly accessible at http://brandon.guggenheim.org. Courtesy of the Solomon R. Guggenheim Museum.

to migrate the ASP scripts, which have become defunct and as a technology pose a security concern, to the programming language Python. Python is widely used and supported for web scripting.

2.1.3. Significance of the artist's or programmer's coding style

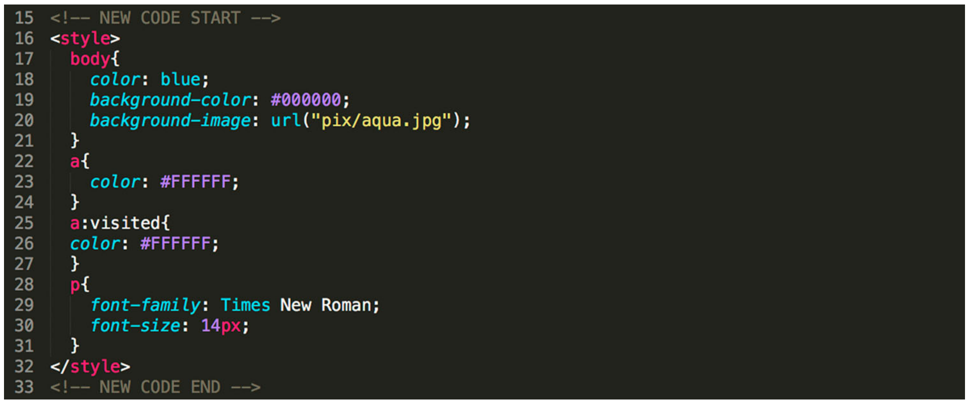
Shu Lea Cheang's web artwork *Brandon* (1998–1999), which was commissioned by the Guggenheim in 1998 and restored in 2016/2017 as part of the current CCBA initiative (Guggenheim Blogs 2017), is a sprawling co-authored web project that tells the tragic story of trans-man Brandon Teena and hosts a wealth of related stories, resources and interactive features for public engagement online (Figure 5). *Brandon* was created using a range of technologies and languages, including Java Applets, Hypertext Markup Language (HTML), JavaScript, Perl scripts and a MySQL database.

Within its HTML, *Brandon* features a particular choice of coding style: the font sizes, colors and effects as well as page layouts were all specified within the HTML of the web pages (Figure 6) and not captured in separate files as is commonly done today using cascading style sheets (CSS). CSS is a more efficient technology and supports uniformity for a website in its "look and feel" as it supports the application of style rules across many pages of content. If the layout and formatting is done without CSS and inside the HTML, larger amounts of coding time are required to implement individual code changes across the site. Although CSS was available when *Brandon* was created, its programmers decided against its use, possibly because the technology was nascent at the time or perhaps because styling uniformity across *Brandon* was not an aesthetic priority for the artist and her collaborators. As a consequence, during the 2016/2017 restoration of *Brandon*, necessary repairs to



```


<TITLE>GARLAND</TITLE>


</HEAD>
<BODY TEXT="blue" LINK="#FFFFFF" VLINK="#FFFFFF" BGCOLOR="#000000"  BACKGROUND="pix/aqua.jpg">


<font size=4>


```

Figure 6. Screenshot of the original source code for Shu Lea Cheang's *Brandon*: The font color, link colors and background are specified within HTML. Courtesy of the Solomon R. Guggenheim Museum.



```
<!-- NEW CODE START -->
<style>
  body{
    color: blue;
    background-color: #000000;
    background-image: url("pix/aqua.jpg");
  }
  a{
    color: #FFFFFF;
  }
  a:visited{
  color: #FFFFFF;
  }
  p{
    font-family: Times New Roman;
    font-size: 14px;
  }
</style>
<!-- NEW CODE END -->
```

Figure 7. Screenshot of the 2016/2017 restoration code for Shu Lea Cheang's *Brandon*: To restore font sizes and colors in keeping with the original style of coding, style elements are added to each HTML file without external CSS pages. Courtesy of the Solomon R. Guggenheim Museum.

its layout and font styling followed the original coding style and were executed within the HTML (Figure 7) instead of introducing modern-day CSS (Engel et al. 2018).

2.2. Informed choice of treatment methods

The conservation professional must strive to select methods and materials that … do not adversely affect cultural property or its future examination, scientific investigation, treatment, or function. (AIC 1994, VI)

As these examples illustrate, recognizing the unique characteristics of an artist's code determines whether that code can be considered replaceable in the event of dysfunction and deprecation, or whether it should be considered preservation-worthy.

Even if source code intervention (e.g. as part of a migration from one programming language to another) is determined necessary to restore an artwork, treatment options can range from minimal intervention to complex restoration. The extent, nuance, and effect of the invasive treatment should and can be a result of deliberate and ethically informed conservation planning. To minimize the degree of code intervention, for example, treatment planning can privilege the choice of a more modern version of the original programming language over replacing it with an entirely different programming language. Secondly, if the migration to a different programming language cannot be avoided, a preferred destination language would be a language similar in syntax and structure to the original; for example, original Java code is better translated to JavaScript than to Python, because the latter is very different in syntax and structure. A migration to Python would necessitate many more programming decisions that could distance the restoration code from the original expression, tone and style of coding. For the 2016/2017 restoration of *Brandon*, it was decided to replace the Java applets, which are no longer supported by modern browsers, with GIF images and JavaScript. GIF images and JavaScript were not only chosen because they are supported by modern browsers, but also because they are already found in other areas of *Brandon* and are therefore native technologies to the work (Guggenheim Blogs 2017).

As is common practice in other areas of art conservation, an informed choice of treatment methods calls for case-by-case decision-making (Appelbaum 2007). This remains true for software-based artworks, even if recent initiatives promote the batch treatment of groups of digital objects in a uniform way, for example by executing multiple artworks with shared software and hardware dependencies in the same emulated legacy environment (Rechert, von Suchodoletz, and Welte 2010; von Suchodoletz, Rechert, and van der Werf 2012; Espenschied et al. 2013; Rechert, Falcao, and Ensom 2016; Espenschied 2017). This approach can be especially relevant, where high quantities of complex digital objects have to be maintained, for example in the context of library and archival collections. However, for art conservation purposes, the application of one default treatment method (such as emulation) for groups of artworks may not always satisfy a conservator's requirement to truthfully reinstate an individual artwork's defining properties, such as its colors, speed, movement or other behaviors.

Case-by-case treatment planning is critically informed by the previous identification of these work-defining properties: "Before any intervention, the conservation professional should make a thorough examination of the cultural property and create appropriate records" (AIC 1994, Guideline 25). A critical examination method for code-based works is source code analysis; human observation alone cannot sufficiently identify or quantify many potential behaviors, such as open durations, randomization or generative features. Furthermore, functions that are already compromised or defunct would no longer be discernible to the viewer, but can be identified by reading the source code. For example, source code analysis of Shu Lea Cheang's *Brandon* revealed deprecated <BLINK> tags in the HTML code that used to make individual words in *Brandon* blink. In contemporary browsers, which no longer recognize <BLINK> tags, these words were simply displayed as static fonts; without analyzing the code, the missing blinking behavior would not have been detected by the viewer.

When planning a software migration, the sustainability of a destination technology or language should be considered, similar to the consideration of the stability and aging properties of an adhesive prior to applying it to a work of art. To support the maintenance of the work and to ensure that future treatments are not inhibited, the selected technology and programming language, including its libraries, and the format types of media assets, should be widely used, supported, understood, and documented. Programmers carrying out a treatment should refrain from introducing obscure or custom solutions that will likely require custom interventions or migrations in the future.

Informed treatment planning should also address anticipated changes in technology. In the case of the *Brandon* restoration, for example, research identified legacy HTML tags in the artwork that were still functioning at the time of the 2016/2017 restoration, but were scheduled to be deprecated within the next 5 years. These HTML tags were then migrated as part of this treatment to prevent the breakdown of their functionality in the near future.

2.3. The supervision challenge: delegating treatment, but not the responsibility

> The conservation professional shall practice within the limits of personal competence and education … (AIC 1994, IV.)
> The conservation professional is responsible for work delegated to other professionals … Work should not be delegated or subcontracted unless the conservation professional can supervise the work directly … (AIC 1994, Guideline 8)

Conservators who are responsible for the care of software- and computer-based art have rarely received training in reading and writing code. Therefore, they commonly have to delegate the examination and treatment of these works to other professionals. Supervising this delegated work can prove extremely challenging, if the conservation professional has little understanding of a programmer's treatment approach. For example, while treatment choices to retain all or parts of the original code or technology seem intuitive to conservators, they may collide with established principles in programming and computer science, as conservators and programmers value and evaluate source code in different ways. To guarantee a conservation-minded decision-making process and desired treatment outcome, it is key for professionals in both disciplines to understand each others approaches and collaborate effectively.

2.3.1. Different approaches between disciplines

Conservators are inclined to look for significances in the artist-selected technology, programming language, and artist's coding style to determine treatment strategies. In contrast, programmers judge code primarily by its quality. For example, programmers will often judge code by its elegance (whether it is well written); whether it is concise (as opposed to verbose); whether it makes best use of the hardware it is intended to run on; and whether the code is clearly written and appropriately annotated. In computer programming, supervisors and colleagues will often conduct code reviews to evaluate and assess the quality of code while it is under development and upon completion as a way to assist programmers and to give feedback. This is analogous to having a professional editor review a manuscript for publication. When programmers strategize code intervention to treat defunct code, their prime criterion in choosing a technology or programming language is not necessarily its long-term sustainability or likeness to the original code or language, but the ability of the technology or language to offer a smooth and efficient execution of the desired functions within a reasonable development timeframe. As different programming languages are designed to serve different purposes and support different functionalities, the quantitative and qualitative efficiency of expression can be higher in one language over another.

For artwork examinations in this research, computer scientists and programmers had to learn to read and evaluate code through a conservator's eyes, i.e. to respect the unique style and character of the code as it was written by the artist regardless of the quality of the code itself. Programmers collaborating with conservators on the treatment of these artworks had to be reassured that they would not be judged primarily by the quality, programming elegance and efficiency of the finished restoration code according to standards that they are accustomed to outside of the art world, but rather by their ability to interpret and treat a work of software-based art in a way that is respectful of and faithful to its original characteristics, i.e. not just in regard to discernible behaviors of the work, but to its underlying technologies.

2.3.2. Developing common ground between disciplines

When planning a conservation treatment, the concerns and preferences of both disciplines, conservation and computer programming, have to be balanced and the advantages of each possible solution have to be weighed against its disadvantages. To avoid misunderstandings and to build a shared language, both sides must commit to mutual teaching and ongoing exchange.

Cross-disciplinary communication in this research was cultivated through formal and informal exchange: At the beginning of each academic semester, computer science students new to the conservation field were introduced to conservation principles, and at the end of each semester, the students and programmers formally presented and discussed their examination findings and treatment progress with an audience of conservators, curators, and other collection care professionals. In turn, the computer scientists introduced media conservators to concepts common to their field, with discussions that guided all phases of the case studies. All programming treatments were conducted in the Guggenheim media conservation lab, which allowed conservators and computer scientists to work together closely and in real time, to make decisions collaboratively and to ensure proper treatment documentation.

A prime tool for making informed treatment choices collaboratively has been prototyping and comparing different possible programming solutions together. The approach of prototyping is analogous to micro-testing the efficiency of a proposed solvent or consolidant in the treatment planning of an artwork; microscopic areas of the artwork that represent certain treatment challenges are selected for testing. For software-based art, the relevant code is copied into a separate file outside of the artwork and amended or migrated. Building prototypes, e.g. in different programming languages, allows programmers to test their ideas and serves as a foundation for discussion with conservators and artists to devise an appropriate treatment plan.

Particular attention should be given to the assessment of these prototypes. Complementary to visual A-B comparisons, performance evaluations and other crucial qualitative assessments, programmers can write code to measure and evaluate results quantitatively, including the speed of execution and other aspects of an artwork that cannot be identified or measured by human observation.

Only through excellent communication and close collaboration between conservators and programmers is it possible to apply the same conservation standards to the examination and treatment of software-based works as is established for other works of art.

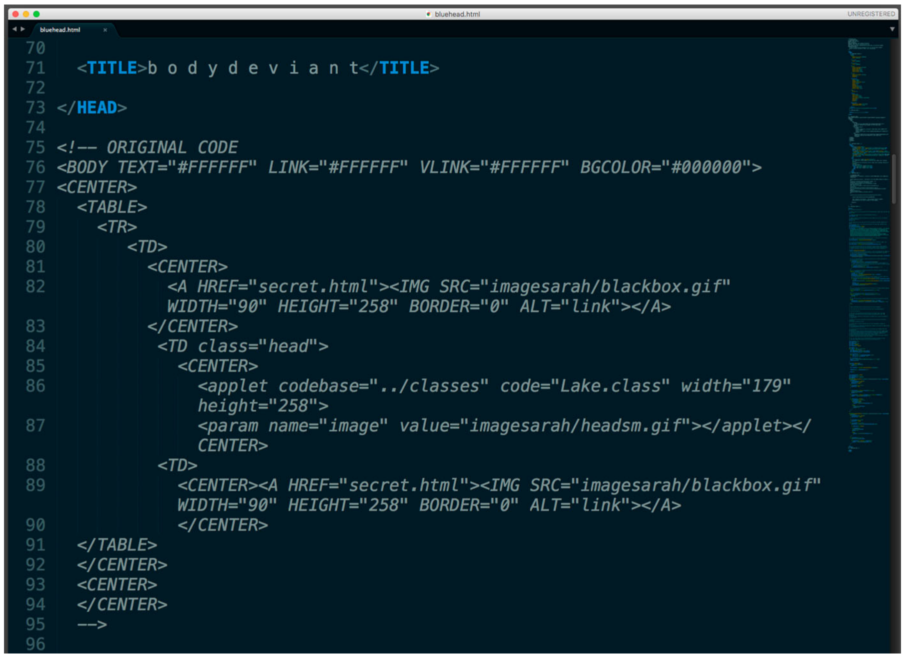
2.4. Compensation for loss: detectability and reversibility

Any intervention to compensate for loss should be documented in treatment records and reports and should be detectable by common examination methods. Such compensation should be reversible and should not falsely modify the known aesthetic, conceptual, and physical characteristics of the cultural property, especially by removing or obscuring original material. (AIC 1994, Guideline 23)

Conservators strive to compensate for “physical loss to the material of a cultural property” or its “loss of original appearance” and to restore its “structural stability, visual unity, and/or function and use” (AIC 1994, Guideline 23, Commentary). Any compensatory measure should be detectable and reversible; detectable to prevent a viewer’s deception of the condition and value of a restored work, and reversible because other treatments may become necessary or favored in the future, and the risk of damage should be minimized. (While the practical feasibility of treatment reversibility has been much debated in the world of tangible cultural heritage, it is quite achievable in the treatment of digital artworks, where the restoration should be carried out on a digital, exact copy of the work.)

The 2016/2017 restoration of *Brandon* served as a case study to develop measures that would enable the detectability and reversibility of compensatory source code intervention. First and foremost, defunct original code in *Brandon* was never deleted. Regardless of whether an artwork is migrated to a different technology and programming language, or if the programming language is maintained and code amendments take place within the original code, a new copy and version should be created to preserve the original or previous version of the artwork. Thereby, every treatment becomes fully reversible. Furthermore, in the new restoration version, defunct or deprecated original code should not be deleted whenever possible. Instead, it should simply be deactivated by “commenting it out,” i.e. by inserting human-readable code annotations that deactivate the execution of lines or blocks of code and label original code (Figure 8). To reverse treatments within the restoration version, the newly added code can also be deactivated by commenting it out, and the old code can be reactivated by deleting the restoration comments.

The method of code annotation also serves the purpose of making a treatment detectable. During the treatment of *Brandon*, headers were created for each of the treated files that specified the restoration project, author, date and purpose of intervention, and comments before and after new blocks of code were added to indicate “NEW CODE START” and “NEW CODE END” (Figure 9).



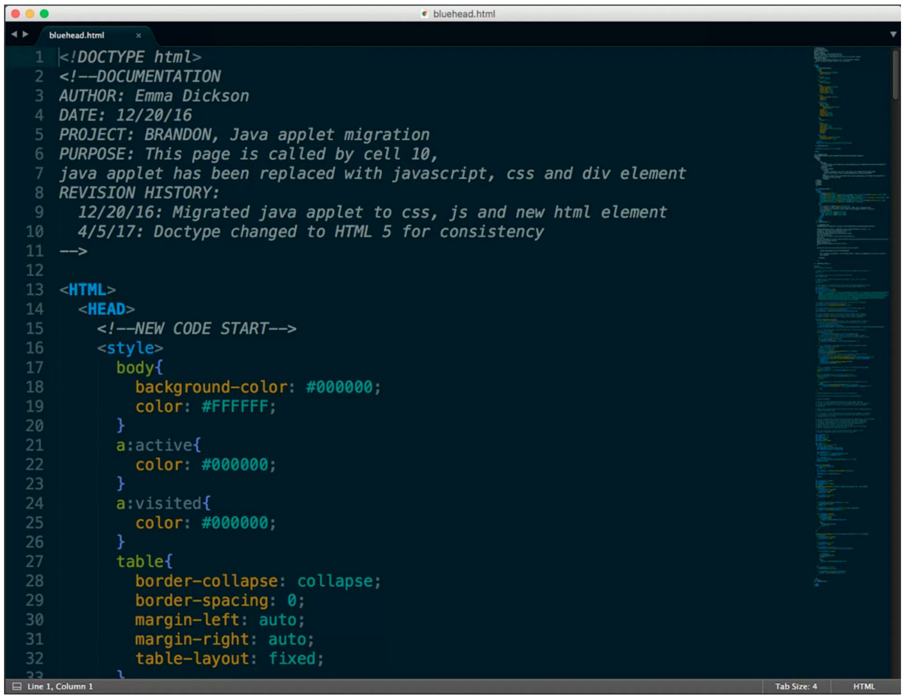
```
  <TITLE>b o d y d e v i a n t</TITLE>

</HEAD>

<!-- ORIGINAL CODE
<BODY TEXT="#FFFFFF" LINK="#FFFFFF" VLINK="#FFFFFF" BGCOLOR="#000000">
<CENTER>
  <TABLE>
    <TR>
       <TD>
         <CENTER>
           <A HREF="secret.html"><IMG SRC="imagesarah/blackbox.gif" WIDTH="90" HEIGHT="258" BORDER="0" ALT="link"></A>
         </CENTER>
          <TD class="head">
            <CENTER>
              <applet codebase="../classes" code="Lake.class" width="179" height="258">
              <param name="image" value="imagesarah/headsm.gif"></applet></CENTER>
          <TD>
            <CENTER><A HREF="secret.html"><IMG SRC="imagesarah/blackbox.gif" WIDTH="90" HEIGHT="258" BORDER="0" ALT="link"></A>
            </CENTER>
  </TABLE>
  </CENTER>
  <CENTER>
  </CENTER>
  -->

```

Figure 8. Screenshot of the 2016/2017 restoration code of *Brandon*. The original but defunct code is not deleted but simply deactivated by commenting it out and labeling it as original code. Courtesy of the Solomon R. Guggenheim Museum.

2.5. Documentation

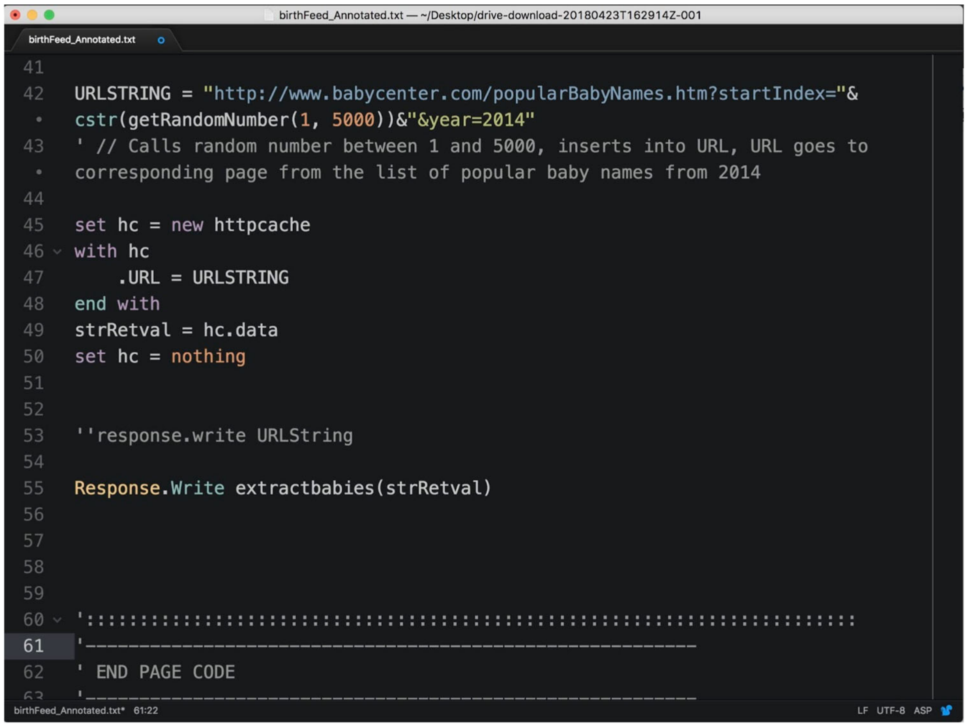
> The conservation professional has an obligation to produce and maintain accurate, complete, and permanent records of examination, sampling, scientific investigation, and treatment. (AIC 1994, Guideline 24)

Previous interdisciplinary research has identified four methods of documenting software-based art: (1) the examiner's annotation of the analyzed source code, (2) narrative documentation, (3) visual illustration such as charts and graphs, and (4) unified markup language (UML) (Engel and Wharton 2015). In this current research, further methods and practices have been developed and tested to create records of artwork examination and treatments.



```
<!DOCTYPE html>
<!--DOCUMENTATION
AUTHOR: Emma Dickson
DATE: 12/20/16
PROJECT: BRANDON, Java applet migration
PURPOSE: This page is called by cell 10,
java applet has been replaced with javascript, css and div element
REVISION HISTORY:
  12/20/16: Migrated java applet to css, js and new html element
  4/5/17: Doctype changed to HTML 5 for consistency
-->

<HTML>
  <HEAD>
    <!--NEW CODE START-->
    <style>
      body{
        background-color: #000000;
        color: #FFFFFF;
      }
      a:active{
        color: #000000;
      }
      a:visited{
        color: #000000;
      }
      table{
        border-collapse: collapse;
        border-spacing: 0;
        margin-left: auto;
        margin-right: auto;
        table-layout: fixed;
      }
```

Figure 9. Screenshot of the 2016/2017 restoration code of *Brandon*. A header in the file identifies the author, date, project and purpose of the code intervention. Courtesy of the Solomon R. Guggenheim Museum.


birthFeed_Annotated.txt — ~/Desktop/drive-download-20180423T162914Z-001
birthFeed_Annotated.txt
birthFeed_Annotated.txt* 61:22
LF UTF-8 ASP

```
41
42  URLSTRING = "http://www.babycenter.com/popularBabyNames.htm?startIndex="&
•   cstr(getRandomNumber(1, 5000))&"&year=2014"
43  ' // Calls random number between 1 and 5000, inserts into URL, URL goes to
•   corresponding page from the list of popular baby names from 2014
44
45  set hc = new httpcache
46  with hc
47      .URL = URLSTRING
48  end with
49  strRetval = hc.data
50  set hc = nothing
51
52
53  ''response.write URLString
54
55  Response.Write extractbabies(strRetval)
56
57
58
59
60  ':::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
61  '---------------------------------------------------------
62  ' END PAGE CODE
63  '---------------------------------------------------------
```

Figure 10. Screenshot of Siebren Versteeg's "birthfeed" ASP scripts. In line 43, the examining programmer explains how the code calls for the baby names. Courtesy of the Solomon R. Guggenheim Museum.

2.5.1. Creating meaningful examination records

When examining the source code of a software-based artwork, a copy of the original code is created. The examining computer scientist or programmer annotates this "examination version" with comments in the code that explain how individual lines or blocks of code define rules, functions and behaviors of the artwork (Figure 10). The comments may indicate, for example, whether code has become defunct or deprecated and is no longer executed; whether it has never been active and may constitute an early, abandoned draft of an idea; whether it has been adopted from previous or other projects; or whether it was coded by different collaborators. Previous research concludes that "source code and the comments in the source code are the most important documentation" both for software engineers and the museum (Engel and Wharton 2015). And indeed, source code annotation as a documentation method offers the most granular record of examination and can be critical for future programmers to understand and interpret an artwork's code, especially if the programming languages used are no longer supported or commonly known.

While source code annotation was also integral to the examination of case studies in this research, concern arose that not all target audiences could access or interpret the documentation inside the code. Since decision-making about the variability and permissible change and treatment of a software-based artwork still lies with conservators, curators and other collection stewards, who are commonly code-illiterate, an additional documentation method was sought that digests and interprets programmers' findings and offers higher-level access to information that is critical to collections care.

To achieve this goal, a worksheet known as the Identity Report, which was developed at the Guggenheim to document time-based media (Phillips 2015), was expanded and adjusted to capture software- and computer-based works in particular. This documentation includes an art historical and technical contextualization of the artwork; the work's anatomy and constituents in relationship to each other; the intended behaviors and functions; its installation and display parameters; the work's aesthetic, conceptual and functional software and hardware dependencies; information about the development environment and history; the history of post-completion modifications and change; the condition and risk assessment, and the short, mid and long term prognosis for necessary maintenance.

To complement this written and illustrated documentation, a further documentation method proved extremely valuable in this study: narrated screen recording. Screen recordings, or video screen captures, are digital recordings of the audiovisual computer screen output that can be accompanied by the examiner's live audio narration. This documentation method served a variety of examination goals, e.g. to create an audiovisual

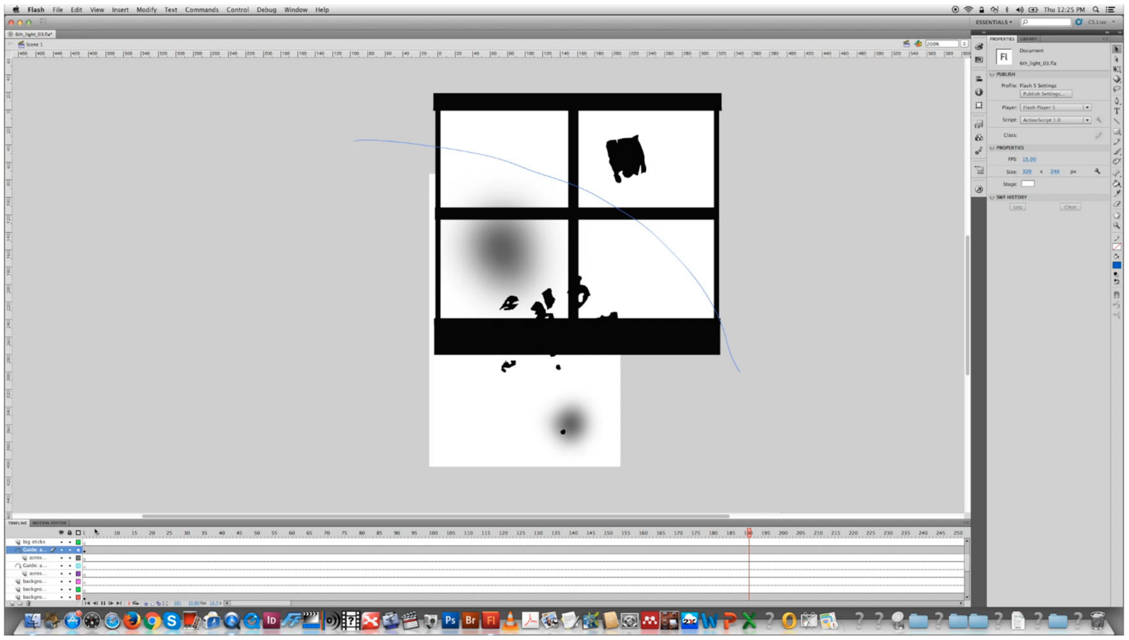


Figure 11. Screenshot of Paul Chan's *6th ~~Light~~* viewed in its native production environment Adobe Flash. Courtesy of the Solomon R. Guggenheim Museum.

reference of an artwork's intended behaviors in its native display environment, including color rendering, movements and speed; to document an artwork's interactive on-screen behaviors; to create an audiovisual record of condition issues such as missing functions, and to document a work's genesis in its native production environment. One example is the creation of a narrated navigation of Paul Chan's *6th ~~Light~~* in Adobe Flash. The examining computer science student navigated through the Flash project while explaining how the artist created large libraries of custom shapes, arranged a selection of these shapes on a complex timeline, animated the movement of the shapes passing the window cross and created the effect of dusk and dawn (Figure 11).

2.5.2. Documentation of conservation treatments

> During treatment, the conservation professional should maintain dated documentation that includes a record or description of techniques or procedures involved, materials used and their composition, the nature and extent of all alterations, and any additional information revealed or otherwise ascertained. (AIC 1994, Guideline 27)

All treatments of software- and computer-based art, whether they involve the manipulation of original source code or not, can effectively be documented with written and illustrated treatment reports as established in the field of art conservation. As soon as code intervention is part of the treatment, however, additional forms of documentation are called for that offer precise location mapping of these interventions. Artist's source code can consist of hundreds or even thousands of discrete files, each of which can contain hundreds or more of lines of code, and restoration intervention can take place across many of these files. The documentation method of code annotation, as described under 2.4, Compensation for Loss, allows conservators to detect treatment intervention within the code and makes restoration code distinguishable from original artist's code. The searchability of treatments across files, however, depends on the use of controlled vocabularies in these code annotations. In the 2016/2017 restoration of *Brandon*, great focus was placed on standardizing annotation terminology and thereby enhancing the searchibility of restoration code. However, as long as this terminology is not standardized within the greater field of conservation, future conservators or programmers would not know the keywords to search for when looking for previous treatments.

A solution to this problem is the parallel use of version control software, for example the application Git (Git Website n.d.). Version control systems are essential in the software development industry and are used to track changes in computer files and coordinate work on these files by multiple programmers. All code changes that are submitted to Git are automatically stamped with the time and the author. The programmer can add decision-making rationale for each coding step, and other members of the conservation team can review the progress by running the software and flagging behavioral bugs that require corrections in the code. Changes to the code are easily searchable and accessible through Git's graphical user interface and the old original and new restoration code can be easily compared through split screen functions. Git also supports the parallel exploration of multiple coding solutions while allowing all of the participants to work on the most current version of the source code.

At the Guggenheim, Git was already in use by the IT and museum website departments, so *Brandon's* treatment documentation could easily be added to the Guggenheim Git implementation. The institutional integration of these insular conservation records seemed crucial to minimize the risk of loss.

As a final measure of treatment documentation, the method of screen recording (see 2.5.1) again proved extremely valuable. It was used to document *Brandon's* audiovisual and interactive behaviors for the before, during and after treatment states. In *Brandon's* case, the decision was made to also create a narrated post-treatment screen navigation of the entire website to provide scholars and general audiences with a full overview of the historic work, as its navigation is not intuitive for many contemporary users who have reported to miss large parts of the complex work. This narrated screen navigation can be found online (Guggenheim Blogs 2017).

2.6. Preventive conservation

The conservation professional should recognize the critical importance of preventive conservation as the most effective means of promoting the long-term preservation of cultural property. (AIC 1994, Guideline 20)

Preventive conservation aims to create appropriate storage, access, and display conditions for artworks to prevent their deterioration or damage, and the need for invasive conservation intervention: "In 'preventive' (or 'indirect') preservation treatments, the object is not touched, but left in a different environment" (Muñoz Viñas 2005, 22). Applied to software- and computer-based art, obvious measures of preventive conservation include secure, redundant and access-regulated server storage. In addition, proactive measures are necessary to create an environment for these works that enables their future access, examination, and treatment.

As with other types of contemporary art, the acquisition process of software- and computer-based artworks is a key moment for caretakers to collect relevant components and information to ensure their sustained collection life. Obtaining the uncompiled, human-readable source code in addition to the compiled, machine-readable executable file that is needed to run the artwork (the equivalent of an exhibition copy in video art) is perhaps the most crucial act of preventive conservation. It became evident in this research that the source code is not only essential for the examination and future treatment of a work, but a key record of self-documentation. To a code-literate person, the code acts like a score that precisely reveals the rules and behaviors of an artwork, even if it fails to run, e.g. due to hardware breakdown or software obsolescence.

If the museum is unable to obtain the uncompiled source code, it could in some cases extract the code by decompiling the artist-provided executable. However, retrieving the source code through decompilation has a number of disadvantages. Firstly, any comments in the code, such as the artist's or programmer's annotations and any unused code, are irretrievably lost in the compilation process and not present in the decompiled code. Secondly, recompiling previously decompiled code, e.g. if the source code needs treatment such as code amendment or migration, can lead to a risk of shifts in the intended artwork behaviors or functionality. And last but not least, decompilation requires decompiler software that is specific to the programming language and version (as well as the operating system in some cases) and also becomes obsolete and likely unavailable over time. This is true for two artworks, which the Guggenheim acquired in 1989 and 1999 as executables only, without their source code. As no appropriate decompilers could be found during this research, it was impossible for the computer scientists to decompile the executables. As a result, the defining behaviors of both works could not be identified and documented. This example shows that if compiled executables are acquired, it is advisable to also source the appropriate decompiler and compiler software immediately. To support reconstructive treatments in the future, compiler software should be sourced even if the uncompiled source code is obtained. In addition, copies of integrated media assets such as images, graphics, and videos should be extracted immediately upon acquisition and stored outside of the artwork. Caretakers should also source any artwork-supporting software, including the required software and code libraries, as soon as the work enters the collection; in the case of the above-mentioned examples by Paul Chan and Siebren Versteeg, this means the legacy versions of Adobe Flash and Macromedia Director.

At the time of acquisition, a full source code analysis is not always possible; basic intake-documentation, however, should include the identification of the work's programming languages, its relevant software libraries and technologies, and its hardware and software environments including the operating system.

An additional valuable historic reference for future conservators, curators, and programmers is created by recording the artwork running in its native display environment at the point of acquisition, when all functions and behaviors are still intact as intended by the artist.

3. Conclusion and outlook

This research has shown that core conservation principles established in the AIC's *Code of Ethics* and *Guidelines for Practice* provide useful guidance for the practice development in the emerging field of software- and computer-based art conservation. This study demonstrated further that treatment planning of these works should not just be informed by an artwork's discernible behaviors, but by functional, conceptual, and other significances identified in the underlying, original source code, such as the artist's choice of technologies, programming languages, and coding style.

As a product of this cross-disciplinary collaboration, practical developments include the implementation of code annotation to make treatment interventions detectable and reversible, and the introduction of new documentation methods, for example screen recording and version control software, to capture code-based artworks and their conservation treatments. As proactive measures of preventive conservation, the authors advise on components and information to collect when acquiring software- and computer-based art.

Among the six investigated conservation principles, one notion that remains most challenging for implementation is the requirement of conservation professionals to supervise delegated conservation work. In the case of software- and computer-based work, the specialist expertise may be so removed from current conservation skillsets that supervision is not easily realized. Cross-disciplinary communication can be especially challenging, as conservators and computer scientists often use different terminology and gravitate towards different treatment approaches. In this research, it has been key for both disciplines to learn about each others' principles, priorities, and decision-making processes over the course of several years. But without this commitment to develop a mutual language, conservation professionals may currently not be able to communicate and collaborate effectively with computer scientists and programmers to ensure that conservation ethics inform and guide the examination and treatment of these works.

With an increasing volume of contemporary art being produced in software- and computer-based media, and more of these works entering collections, the authors recommend that art conservation programs respond to the increasing need for specialized expertise and adjust their curricula and pre-program requirements accordingly. Ideally, conservators of software- and computer-based art would combine training in programming (e.g. by means of an undergraduate degree in computer science) with a graduate degree in art conservation. To enable conservation supervision of delegated programming work, emerging conservators of contemporary art should be trained within their curriculum to understand basic principles and approaches in computer programming. Similar interfaces between conservation and other disciplines, such as art history or conservation science, are already being cultivated in conservation training. Only by expanding this cross-disciplinarity can the field of art conservation embrace the next generation of contemporary art production.

Acknowledgements

The authors would like to thank the artists Paul Chan, Siebren Versteeg, and Shu Lea Cheang for their support of this research. Further thanks go out to all current and former supporters of and contributors to the CCBA initiative and Guggenheim-NYU research collaboration since 2014: At the Solomon R. Guggenheim Museum we would like to thank Lena Stringari, Deputy Director and Chief Conservator; Jonathan Farbowitz, Fellow for the Conservation of Computer-based Art; Alexandra Nichols, former Kress Fellow for Time-based Media Conservation; Brian Castriota, former Samuel H. Kress Fellow in Time-based Media Conservation; Amy Brost, former Andrew W. Mellon Graduate Intern for Time-based Media Conservation; Lia Kramer, former Polonsky Intern for Digital Humanities, and Jiwon Shin, former summer intern for Time-based Media Conservation.

At New York University's Courant Institute for Mathematical Sciences, particular thanks go to the computer science students and graduates who contributed to the source code analysis, treatment prototyping and treatment of Guggenheim collection works: Jiwon Shin, Aarti Chandrakant Bagul, Shan Shao, Jiwon Shin, Michelle Liu, Vivian Peng, Emily Hua, Caroline Slason, Mia Matthias, Emma Dickson, Jillian Zhong, Kaitlin Gu, Justin Pak, Karl Toby Rosenberg, Yonatan Medina, Alejandra Trejo Rodriguez, Emily Fong, and Xincheng Huang.

The CCBA initiative at the Guggenheim would not be possible without the generous support by the Carl & Marilynn Thoma Art Foundation, the New York State Council on the Arts with the support of Governor Andrew Cuomo and the New York State Legislature, Christie's, and Josh Elkes.

We would further like to thank our colleagues who took time to review this publication: Professors Ernest Davis and Craig Kapp, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University; Prof. Glenn Wharton, Department of Museum Studies, New York University; Lena Stringari and Jeffrey Warda, Conservation Department, Solomon R. Guggenheim Museum; and Agathe Jarczyk, Department of Conservation-Restoration, Bern University of the Arts.

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Deena Engel is a Clinical Professor in the Department of Computer Science at the Courant Institute of Mathematical Sciences of New York University as well as the Director of the Program in Digital Humanities and Social Science. She teaches undergraduate computer science courses on web and database technologies, as well as courses for undergraduate and graduate students in the Digital Humanities and the Arts. She also supervises undergraduate and graduate student research projects in the Digital Humanities and the Arts, and collaborates on research on the conservation of software-based art. She received her master's degree in comparative literature from SUNY-Binghamton and her master's degree in computer science from the Courant Institute of Mathematics at New York University, NY, USA.

Joanna Phillips is the Senior Conservator of Time-based Media at the Solomon R. Guggenheim Museum in New York, where she founded the media art conservation lab in 2008. At the Guggenheim, Phillips has developed and implemented new strategies for the preservation, reinstallation, and documentation of time-based media works. Phillips publishes and lectures on this topic internationally. She founded and heads the Guggenheim's ongoing initiative Conserving Computer-based Art (CCBA) and is a founding co-organizer and multiple host of the EMG conference series TechFocus. Phillips also co-organized the international symposium "Collecting and Conserving Performance Art" in Germany. Prior to her Guggenheim appointment, Phillips specialized in the conservation of contemporary art at the Swiss Institute for Art Research in Zürich and explored the challenges of media art conservation as a research conservator in the Swiss project "AktiveArchive." Phillips holds a master's degree in paintings conservation from the Hochschule der Bildenden Künste in Dresden, Germany.

ORCID

Deena Engel https://orcid.org/0000-0003-2703-3075
Joanna Phillips http://orcid.org/0000-0002-2923-8827

References

AIC (American Institute for Conservation of Historic and Artistic Works). 1994. Code of Ethics and Guidelines for Practice. Accessed August 1, 2018. http://www.conservation-us.org/ethics.

Appelbaum, B. 2007. *Conservation Treatment Methodology*. Oxford: Elsevier Butterworth-Heinemann.

Appelbaum, B. 2011. "Conservation in the 21st Century: Will a 20th Century Code of Ethics Suffice?" In *Ethics & Critical Thinking in Conservation*, edited by Pamela Hatchfield, 1–10. Washington, DC: American Institute for Conservation of Historic & Artistic Works.

Ashley-Smith, J. 2017. "A Role for Bespoke Codes of Ethics." In *ICOM-CC 18th Triennial Conference 2017 Copenhagen, Preprints*, edited by Janet Bridgland, 1–10. Paris: ICOM.

Chan, P. 2016. Paul Chan, Artist interview by Joanna Phillips, Deena Engel, Lauren Hinkson, Kaitlin Gu, Jillian Zhong, and Amy Brost at the artist's Badlands Studio, New York, NY, USA, March 11, 2016. Audio recording, Conservation Department Records, Solomon R. Guggenheim Museum.

Clavir, M. 2002. *Preserving What is Valued: Museums, Conservation, and First Nations*. Vancouver: UBC Press.

Dekker, A., and P. Falcao. 2016. Interdisciplinary Discussions about the Conservation of Software-Based Art. A report written by PERICLES Project partners in collaboration with participants of the Community of Practice (CoP) on Software-Based Art. Accessed August 1, 2018. http://pericles-project.eu/uploads/files/PERICLES_SBA_CoP_report_2017_FINAL.pdf.

ECCO (European Confederation of Conservator-Restorers's Organisations). 2003. E.C.C.O. Professional Guidelines (II) – Code of Ethics. http://www.ecco-eu.org/fileadmin/user_upload/ECCO_professional_guidelines_II.pdf (accessed December 28, 2018).

Engel, D., L. Hinkson, J. Phillips, and M. Thain. 2018. "Reconstructing Brandon (1998–1999): A Cross-disciplinary Digital Humanities Study of Shu Lea Cheang's Early Web Artwork." *Digital Humanities Quarterly* 12 (2). Accessed December 28, 2018. http://digitalhumanities.org/dhq/vol/12/2/000379/000379.html.

Engel, D., and G. Wharton. 2014. "Reading Between the Lines: Source Code Documentation as a Conservation Strategy for Software-Based Art." *Studies in Conservation* 59: 404–415.

Engel, D., and G. Wharton. 2015. "Source Code Analysis as Technical Art History." *Journal of the American Institute for Conservation* 54: 91–101.

Espenschied, D. 2017. Preservation by Accident is not a Plan: Vint Cerf and Dragan Espenschied in Conversation. Accessed August 1, 2018. http://rhizome.org/editorial/2017/may/30/preservation-by-accident-is-not-a-plan/.

Espenschied, D., K. Rechert, I. Valizada, D. von Suchodoletz, and N. Russler. 2013. "Large-Scale Curation and Presentation of CD-ROM Art." *iPRES* 2013: 1–8. Accessed August 1, 2018. http://purl.pt/24107/1/iPres2013_PDF/Large-Scale%20Curation%20and%20Presentation%20of%20CD-ROM%20Art.pdf.

Git Website. n.d. Accessed August 1, 2018. https://git-scm.com.

Guggenheim Blogs. 2016. How the Guggenheim and NYU are Conserving Computer-Based Art—Part 1. Blog entry by Caitlin Dover, October 26, 2016. Accessed August 1, 2018. https://www.guggenheim.org/blogs/checklist/how-the-guggenheim-and-nyu-are-conserving-computer-based-art-part-1. How the Guggenheim and NYU Are Conserving Computer-Based Art—Part 2. Blog entry by Caitlin Dover, November 4, 2016. Accessed August 1, 2018. https://www.guggenheim.org/blogs/checklist/how-the-guggenheim-and-nyu-are-conserving-computer-based-art-part-2.

Guggenheim Blogs. 2017. Restoring Brandon, Shu Lea Cheang's Early Web Artwork. Blog entry by Phillips, J., Engel, D., Dickson, E. and J. Farbowitz, May 16, 2017. Accessed August 1, 2018. https://www.guggenheim.org/blogs/checklist/restoring-brandon-shu-lea-cheangs-early-web-artwork.

Guggenheim Collection Online. n.d. Chan. Paul Chan, 6th. Accessed August 1, 2018. https://www.guggenheim.org/artwork/20625.

Guggenheim Collection Online. n.d. Cheang. Shu Lea Cheang, Brandon, Accessed August 1, 2018. https://www.guggenheim.org/artwork/15337.

Laurenson, P. 2004. The Management of Display Equipment in Time-based Media Installations. *Tate Papers* No. 3 / Spring 2005. Accessed December 28, 2018. https://www.tate.org.

uk/research/publications/tate-papers/03/the-management-of-display-equipment-in-time-based-media-installations.

Laurenson, P. 2006. Authenticity, Change and Loss in the Conservation of Time-Based Media Installations. Tate Papers No. 6, Autumn 2006. Accessed December 28, 2018. https://www.tate.org.uk/research/publications/tate-papers/06/authenticity-change-and-loss-conservation-of-time-based-media-installations.

Laurenson, P. 2014. "Old Media, New Media? Significant Difference and the Conservation of Software-Based Art." In *New Collecting: Exhibiting and Audiences After New Media Art*, edited by B. Graham, 73–96. Dorchester: Routledge.

Miller, C. 2017. Adobe Announces Plans to Shutter Contribute, Director, and Shockwave Starting Next Month. In *9TO5Mac*, January 28, 2017. Accessed August 1, 2018. https://9to5mac.com/2017/01/28/adobe-shutter-contribute-director-shockwave/.

Muñoz Viñas, S. 2005. "Preventive Preservation." *Contemporary Theory of Conservation*. Burlington: Elsevier Butterworth-Heinemann, 22.

Muñoz Viñas, S. 2009. "Minimal Intervention Revisited." In *Conservation Principles, Dilemmas and Uncomfortable Truths*, edited by A. Richmond, and A. Bracker, 47–59. London: Elsevier Butterworth-Heinemann.

Phillips, J. 2010. "Kunstmaterial oder Elektroschrott? Über das Sterben und Auferstehen elektronischer Kunstwerke." In *Wann stirbt ein Kunstwerk?*, edited by Angela Matyssek, 105–124. Munich: Verlag Silke Schreiber.

Phillips, J. 2012. "Shifting Equipment Significance in Time-based Media Art." *Electronic Media Review* 1. Washington: 139–154.

Phillips, J. 2015. "Reporting Iterations: A Documentation Model for Time-Based Media Art." *Performing Documentation, Revista de História da Arte*, edited by Gunnar Heydenreich, Rita Macedo, and Lucia Matos, 168–179. Lisbon: Instituto de História da Arte (IHA) and the Network for Conservation of Contemporary Art (NeCCAR). http://revistaharte.fcsh.unl.pt/rhaw4/RHAw4.pdf.

Rechert, K., P. Falcao, and T. Ensom. 2016. Introduction to an Emulation-based Preservation Strategy for Software-based Artworks, Tate Research Publications. Accessed August 1, 2018. http://www.tate.org.uk/download/file/fid/105887.

Rechert, K, D. von Suchodoletz, and R. Welte. 2010. "Emulation Based Services in Digital Preservation." Proceedings of the 10th annual joint conference on digital libraries (JCDL '10). ACM, New York, NY, USA, 365–368.

SBMK (Stichting Behoud Moderne Kunst). 1999. "The Decision-Making Model for the Conservation and Restoration of Modern and Contemporary Art." In *Modern Art: Who Cares?*, 164–172. London: Archetype Publications.

The Variable Media Approach. 2003. *Permanence Through Change: The Variable Media Approach*. New York: The Solomon R. Guggenheim Foundation and Montreal: The Daniel Langlois Foundation for Art, Science, and Technology. Accessed December 28, 2018. http://www.variablemedia.net/e/preserving/html/var_pub_index.html.

Van de Vall, R. 2009. Towards a Theory and Ethics for the Conservation of Contemporary Art. Art D'Aujourd'Hui, Patrimoine de Demain, Conservation et Restauration des Oeuvres Contemporaines. 13es journées d'études de la SFIIC. Paris. 51–56.

Van de Vall, R. 2015. "The Devil and the Details: The Ontology of Contemporary Art in Conservation Theory and Practice." *The British Journal of Aesthetics* 55 (3): 285–302.

Versteeg, S. 2014. Artist Interview By Joanna Phillips, Deena Engel, Aarti Bagul, Shan Shao, Jiwon Shin and Brian Castriota at the Guggenheim Media Conservation Lab, New York, October 24, 2014. Audio recording and transcript, Conservation Department Records, Solomon R. Guggenheim Museum.

Villers, C. 2004. "Post Minimal Intervention." *The Conservator* 28 (1): 3–10.

von Suchodoletz, D., K. Rechert, and B. van der Werf. 2012. "Long-term Preservation in the Digital Age – Emulation as a Generic Preservation Strategy." *Praxis der Informationsverarbeitung und Kommunikation* 35 (4): 225–226.

Weyer, C. 2010. "Media Art and the Limits of Established Ethics of Restoration." In *Theory and Practice in the Conservation of Modern and Contemporary Art*, edited by U. Schädler-Saub and A. Weyer, 21–32. London: Archetype Publications.

Wharton, G. 2018. "Bespoke Ethics and Moral Casuistry in the Conservation of Contemporary Art." *Journal of the Institute of Conservation* 41 (1): 58–70.