Guidance for Condition Checking Software-based Artworks

Tom Ensom

Time-based Media Conservation, Tate

Document Last Updated: 12 March 2025

Document Version: 01.00

Contents

1 Document Background

This document is a guide to the condition checking and documentation of software-based artworks, intended primarily for conservators of time-based media artworks. Condition checking for software-based art involves examining the artwork's components and assessing their role, condition and significance. For software-based art condition is considered not only in terms of physical degradation but also the longevity of the technologies in relation to obsolescence. Information gained during condition checking is used to develop strategies for the artwork's long-term care. This information is recorded in and supported by documentation, transmitting knowledge that will support future conservation work and providing a record of the material history of the artwork. This guide was created as part of Tate's Software-based Art Preservation Project, serving as an introduction to the topic for other conservation staff and as a foundation for formalising best practices within the team.

This document was intended for internal use at Tate and is being shared more widely as a standalone document. As it was created primarily for internal use by Tate's Time-based Media Conservation team, it contains references to other internal documentation which may not be clear to those not working within the institution. This PDF represents a snapshot of a live page

on Tate's Time-based Media Conservation Wiki, captured on the date noted at the top of this document. The wiki page will continue to evolve alongside other internal workflows, policies and procedures, while this document will remain a static project output.

2 Introduction

This document provides guidance for condition checking and documenting software-based artworks (i.e. artworks which employ software as their primary artistic medium). Software-based art is highly variable in its constituents and different artworks can demand very different approaches — this page highlights some common considerations. In order to address the needs of a specific software-based artwork, you may need to work closely with the artist, the artists team and/or a technical specialist (e.g. programmer or engineer) during this process.

The purpose of the condition checking and documentation process is multi-faceted, and goals might include:

- Ensuring we have gathered the components needed to support the artwork's life in the collection. This includes those needed to run the original software (e.g. operating systems, computer hardware) and support future treatments (e.g. spare electronic components, source code).
- Understanding the role of the components of the artwork and how they fit together to support its display.
- Gathering and creating documentation that will support the above goals and the artworks life in the collection.
- Understanding the condition of the artwork and identifying treatment requirements.
- For example:
    - Have any components failed?
    - Are any components obsolete?
    - Is a treatment required at point of acquisition or prior to display?

3 Preparation

Before you can begin condition checking, you will need to identify and gather together the components of the artwork. While these are often supplied by the artist or gallery involved, sometimes they may need to be sourced by Tate, typically at the point of acquisition or in preparation for a display. Components associated with software-based artworks are many and varied, but may include:

- Installer, application or binaries for any **custom software** components (i.e. the executable software which is run when the work is displayed).
- **Source materials** for any custom software components (e.g. source code, project files, assets).
- **Computer system** capable of running the software.
- Any **additional hardware** required in addition to the computer system (e.g. display equipment, peripherals, interfaces).
- Installers, applications or binaries for any **'off-the-shelf' software** required to run other components.

You will then need both the time and space to set these components up and test them. If you are inspecting or removing internal components with exposed circuit boards, you will need to set up a workspace which mitigates possible damage through electro-static discharge (e.g. through use of an anti-static mat and wristband combination).

4 Condition Checking and Documentation Prompts

Condition checking and documentation processes will vary depending on the type of system and software involved. If you are not familiar with the technologies used, consult someone in the team who is or consider working with an external specialist. Below are some prompts describing some common steps:

- **Disk imaging**. For any computers or optical media acquired, assess the storage media contained and create suitable disk images (see Disk Imaging). The idea of this process is to create a bit-for-bit backup of the relevant data contained on storage media. It is typically best to do this before powering a computer on for the first time, ensuring that data is safely backed up before risks of alteration are introduced during booting and subsequent use.
- **Check physical condition of computers** inside and outside. Dust build-up should be removed using an air duster and cleaning tools, taking care not to damage any components. Corrosion, failed components (e.g. capacitors) and other potential electrical problems should also be checked for and repairs planned as needed.
- **Testing the software and hardware**. Boot the computer and test that the software is functioning correctly, ideally using reference materials (e.g. video capture) or with a representative of the artist present. If no hardware has been supplied, you will need to source and configure suitable hardware based on existing information (ideally using a system requirements specification supplied by the artist).
- **Disambiguating versions of software**. If there are multiple copies of software components, it is important to understand how and why they differ, so that an informed decision can be made about which to retain. Even if versions are no longer used to display the work, you may want to extract these and create components for them, as this helps support an understanding of the artwork's technical history. You can identify differences through examination, checksum comparison and diffing.
- **Creating and gathering documentation** according to the needs of the artwork. Documentation of software-based art entails use of templates regularly used for other artwork types (e.g. display spec, production diagram) but requirements typically extend beyond what these templates support. The Software-based Art Conservation Report template can be used to guide documentation work or you can use your own approach. Ideally, documentation should provide an overview of the hardware and software used to display the artwork (currently and historically), how these components were created and how they function. Gathering and/or creating more detailed documentation of components provides valuable insights in understanding the artwork and supporting future treatments. Common component types, with examples of useful documents, are listed below:
    - **Hardware component documentation**:
        - **Computers**: maintenance manuals; description of hardware specification, installed operating system and installed software. See Documenting Computer Hardware section for information about specific tools.

        - **Peripheral hardware** (e.g. input/output devices, display equipment): maintenance manuals; relevant information about their hardware/software specification.
        - **Custom hardware**: circuit diagrams; bill of materials (i.e. a list of components used); any other documentation that might be needed to recreate the hardware from scratch.
    - **Software component documentation**:
        - **Custom software**: description of dependency relationships; functional description (i.e. what the software does in plain language); in-line documentation of code (e.g. comments embedded in code).
        - **Complex folder structures** (e.g. project files): file/folder structure listing (e.g. "tree" or Brunnhilde output); list of file formats contained (e.g. DROID output saved as a CSV file in the artwork folder); use to generate a more detailed collection of documentation.
        - **Off-the-shelf software** (e.g. drivers, libraries): third-party manuals and technical specification information; description of dependency relationships (if relevant).
        - **Video and photographic documentation** of the artwork as a whole and/or specific components. Video documentation of dynamic elements via camera or screen capture can be particularly useful where this does not already exist, as it forms a reference for how the work behaves. It may be possible to request this form of documentation be produced by Tate Photography (this was done in the case of e.g. Donald Rodney's Psalms).
- **Complete dedicated TMS fields** using information gained in prior steps (see TMS Software-based Artwork Components).

4.1 Guidance for Documenting Computers

4.1.1 Documenting Windows Computers

4.1.1.1 Hardware Information

A tool such as Microsoft's System Information (part of Windows by default) and the freeware HWINFO can be used to gather extensive information about Windows-based hardware and software environment. Outputs should be saved to file and stored in the artwork folder.

4.1.1.2 Operating System Version

Identify Windows version (including build number) by running the following program from the command-line:

```
ver
```

Identify Windows edition (e.g. Home, Professional, Enterprise) currently activated by running the following script from the command-line:

```
slmgr.vbs -dli
```

4.1.1.3 Installed Software

Generate a text file listing installed programs and their version numbers in Windows 10:

1. Open Command Prompt (be sure to run as Administrator)
2. Enter the following command:

```
wmic
```

3. Enter the following command, replacing the .txt. file name:

```
/output:C:\InstallList.txt product get name,version
```

4.1.1.4 Identifying Software Dependencies

Software can have dependency relationships with other software — it *depends* on that other software in order to operate correctly or at all. For software components of software-based artworks, identifying these is useful as it helps us manage these dependencies during the life of the artwork. Some tools which can help reveal further information about software and hardware dependencies are listed below:

- **Task Manager** is Windows standard and built-in tool for monitoring background activity. This can reveal what programs and services are currently running on a computer, which are set to run on system startup, and how the computer's hardware components are being used.
- **CFF Explorer** is a binary analysis tool, which can be used to retrieve metadata and dependency information from Windows Portable Executable (.exe) files. Dependency information gained cannot be totally relied on, as other executable files may be executed at runtime.
- **DriverView** is a system analysis tool which lists the (.sys) drivers currently loaded by Windows. This can be particularly useful in identifying non-Windows drivers which might have been installed.
- **Sysinternals Process Monitor** and **x64dbg** are process analysis tools which can be used to identify libraries and other resources being accessed by a software program. This can be very useful in identifying how the software program is interacting with its environment e.g. libraries accessed at runtime.

4.1.2 Documenting Mac Computers

4.1.2.1 System Information

The 'About This Mac' and 'System Information' tool in MacOS can be used to gather system information including hardware and software environment information.

1. Click the Apple icon at the top left corner of the screen. Select "About This Mac" from the dropdown list. Some information, such as GPU, can be found on the Overview screen that opens.
2. The model of the motherboard can be found by copying the serial number from the Overview screen and pasting it into Apple's check coverage tool or cross-referencing the model number of the Mac with data on everymac.com.
3. Select "System Report" open the System Information tool, where you can find more detailed hardware information. System Information output can be saved to a file by navigating to File -> Save.

4.1.2.2 System Profiler

As of MacOS 10.14 (Mojave), System Information only outputs in an XML-based .spx format which can be opened on other Macs but is hard to read on other platforms. An alternative is to use a command-line tool called System Profiler which outputs the same information but can be saved as plain text. Call it by typing:

```
system_profiler
```

Send the output to a plain text file using:

```
system_profiler > output.txt
```

4.1.3 Documenting Linux Computers

Note that this guidance is for Debian-based distributions of Linux such as Ubuntu.

4.1.3.1 Hardware Information

Run the lshw command to print hardware information:

```
lshw
```

Or for a concise version:

```
lshw -short
```

You can save this information to a file by adding a pipe to text onto the command:

```
lshw > lhsw_output.txt
```

4.1.3.2 Installed Software

You can use apt to tell you which packages are installed:

```
apt list --installed
```