Artwork Classifications—PAFDAO Project

Dianne Dietrich

Contents

A — Structure of the classifications

One sentence describing the purpose of the following classification, including the identifier used in PREMIS to assign this classification to an artwork.

Short description

A few sentences describing the properties that define a work that have the stated classification.

Implications for access

This section includes a description of the issues the project team encountered when trying to access a work in this classification. When applicable, there will be a description of possible strategies to consider when providing access to such a work.

Restoration potential

In this section, the project team considers the process by which one might get such a work running on a current system, without the use of an emulator or virtual machine. It is important to note here that the project team did not attempt to restore every work in the collection, and some descriptions in this section are based on an understanding of the properties of the work and external research, rather than first-hand restoration experience.

References

Sources consulted and other recommended reading.

B — Browser-Based Works

This classification defines browser-based works. The PREMIS identifier used is "BROWSER."

Short Description

Browser-based works are defined as those artworks where the user interacts with the work by accessing files through a web browser. While browser-based works use file formats commonly associated with the Internet, they may or may not require Internet connectivity to function properly.

Implications for access

Access to browser-based works can range from simple to complex to access, depending on a number of variables.

In the simplest case, a browser-based work is on a disc that has the standard ISO9660 filesystem and consists solely of files that do not require third-party plugins or specialized auxiliary software (beyond a web browser) to be rendered properly. (Note that it is not required in this case that the only filesystem present be ISO9660, but that at least one of the filesystems present on the disc is in that standard format.) It should be possible to view the work on a current system. It is important to note several complicating factors, even for this simple case.

First, web standards have evolved considerably over time. To take one example, HTML has changed since its first version. Various HTML elements have been deprecated with subsequent versions, meaning that their use is discouraged now.[a] There are also some elements from prior versions that are no longer included in subsequent versions and are no longer considered valid HTML. If these elements exist in HTML files in browser-based works, they may not displayed [1].

If the browser-based work requires third-party plugins or other auxiliary software to render properly, access can get considerably more complicated. It may not be possible to install the preferred plugin or software on a current system, because the installation files are no longer available.[b] Even if it is possible to obtain the specific installer file for the required plugin or software—if, for instance, the artist included it with the work—it may not be possible to install these executable files on a current system. The available newer versions of the required plugins or software may or may not be compatible with the files in the work.

In both of these cases, viewing the work in an emulator or virtual machine that closely matches the stated system requirements of the work may be the optimal way to interact with the work. If no system requirements are explicitly stated, it is recommended to consider which operating systems and web browsers (and versions) were contemporary with the work, and configuring an emulator or virtual machine to closely match that environment.

A further complication arises when the browser-based work is on a disc that only has an HFS filesystem. In this case, the above description of issues still does apply—with respect to plugins and browser versions—but it is also recommended to read the considerations for "HFS File System" (Section F) for additional information.

Finally, there are aesthetic concerns with browser-based works. Styling within browsers has evolved dramatically over the years [2, 3], and the look and feel of a browser-based work may no longer match the artist's vision if viewed on a current system.

Restoration potential

As with access, restoration potential of browser-based works is variable. The process of reworking HTML files or other auxiliary files, such as scripts, so that they no longer contain deprecated or invalid elements may be straightforward or complicated, depending on how much and what kind of revision is needed.

[a]It is worth mentioning here that artists might not have adhered to contemporaneous HTML standards, so that invalid or deprecated elements might have always existed in the work, rather than being a product of aging.

[b]It also might be possible that only a specific version is compatible and others are not.

Browser-based works that include files that require specific third-party plugins or auxiliary software may be more difficult to restore. The reasons for this follow from the difficulties in providing access to works that contain executable files (See Section D). Again, as before, while restoration in this manner may be feasible, the process of restoring a work may alter the aesthetics of the work significantly and this may not be an optimal outcome.

References

[1] *HTML element - Wikipedia, the free encyclopedia*. URL: `http://en.wikipedia.org/w/index.php?title=HTML_element&oldid=628440350#Element_status`.

[2] *The Evolution of the Web*. URL: `http://evolutionofweb.appspot.com/`.

[3] *browsers.evolt.org*. URL: `http://browsers.evolt.org/`.

C — Virtual Reality Components (Parent: Browser-Based Works)

This classification addresses browser-based works that have virtual reality components. The PREMIS identifier used is "VRML."

Short description

Virtual reality components in browser-based works were often built using Virtual Reality Markup Language (VRML), which is now obsolete and has been superseded by X3D. VRML is a file format specifying how to render "3-dimensional interactive vector graphics" [1] with various properties that correspond to real-world attributes of objects. Even though VRML files are plain text, they do require specific software to render properly within a work. The most common file extension for these files is .wrl.

Implications for access

Browser-based works with VRML often require appropriate plugins to run properly. Technically, this means that, as long as one can install the appropriate plugin, it should be theoretically possible to view a browser-based work with VRML components on a current system.

The project team found several complications in running browser-based works with VRML. First, many of the browser plugins that render VRML are no longer supported, have ceased active development, or are no longer available for download on the web. Second, some browser-based works may also require additional, sometimes proprietary and/or also obsolete, plugins that may, in combination, come into conflict with one another on a current system [2].

Given these complications, it may often be easiest to run a browser-based work with VRML on an emulator or virtual machine that is set up to closely match the stated system requirements of the work. Many of these artworks included preferred software and plugins on the disc itself, obviating the need to track down now-obsolete or unavailable software. Additionally, if an artist developed a work with a particular piece of rendering software in mind, it may only render properly in that particular software or there may also be preferred software-specific settings to optimize viewing of the work.

Restoration potential

VRML files are plain text, so it is technically possible to read the code. This does mean that it may be possible for one to re-create the work in another programming language, though this will require a considerable amount of effort and programming expertise.

VRML was superseded by X3D in 2004 [3]. As of this writing (2014), while the project team was able to find some tools that purported to convert VRML files into X3D [4, 5, 6], their efforts were not successful. For instance, when trying the online converter in [5], it was difficult to copy longer .wrl files into the site's web form. Second, the resulting .x3d file did not have any of the color information that the original file had. Since a file-by-file translation proved unsuccessful at this time using one tool, the level of difficulty in restoring a VRML-based work—with multiple interrelated files—is not known.[a]

[a]The project team was unable to try conversion using the other two software mentioned.

References

[1] *VRML - Wikipedia, the free encyclopedia.* URL: `http://en.wikipedia.org/w/index.php?title=VRML&oldid=605021649`.

[2] *Dependency hell - Wikipedia, the free encyclopedia.* URL: `http://en.wikipedia.org/w/index.php?title=Dependency_hell&oldid=625304085`.

[3] *Visualizing the Semantic Web: XML-based Internet and Information Visualization.* URL: `http://books.google.com/books?id=YbWxDNZkiaUC&lpg=PR2&pg=PA127#v=onepage&q&f=false`.

[4] *VRML97 to X3D Translation.* URL: `http://ovrt.nist.gov/v2_x3d.html`.

[5] *instantreality 1.0 - tools - X3D encoding converter.* URL: `http://doc.instantreality.org/tools/x3d_encoding_converter/`.

[6] *MeshLab.* URL: `http://meshlab.sourceforge.net/`.

D — Executables in Works

This classification outlines considerations when executable files are present in works. The PREMIS identifier used is "EXE."

Short description

Executable files are files that can be readily run by a computer [1]. These files are often only compatible with specific hardware and/or operating systems. An artwork may consist of one or more executable files, or may consist of executable file(s) that installs software to a specific operating system. Further, executable files may be present on a work because they are for auxiliary software (a browser plugin, for example) for the work. It should be noted that auxiliary software may not be required for the work to function properly.

Implications for access

The difficulty of accessing artworks with executables can range from relatively straightforward to complex, depending on the hardware and software environment required to run the work. The work may contain PC-compatible executable files, Macintosh-compatible executables, or both, depending on the file systems present on the disc. As noted above, executables files contained within a work may be auxiliary and not required for the work to function properly, so it is important to take that into consideration when evaluating access strategies for such works.

Macintosh-compatible executable files will most likely require an emulator or virtual machine in order to access. Works designed for the Motorola 68k processors [2] and the PowerPC processors [3] could run in Macintosh OS versions up to OS 9. Early versions of OS X did include a compatibility environment, called "Classic Environment" [4, 5] that would allow a user to run programs originally designated for the pre-OS X operating systems. Support for this environment ended with OS 10.4, and the newer Intel-based Macintosh computers (manufactured after mid-2006) will not support the "Classic Environment." Programs designed for OS X for a PowerPC-based Macintosh could be run using the Rosetta [6] software; however, this software only ran on OS 10.4 through 10.6. As of OS 10.7, Rosetta is not supported, and there is no way to run PowerPC programs on a machine running 10.7 or above without using an emulator or virtual machine.

PC-compatible executable files may or may not require an emulator or virtual machine in order to access. Works that have 16-bit executable files designed to run in Windows 3.1 can run in Windows 95, Windows 98, and Windows ME natively. A compatibility layer, "Windows on Windows" allows 16-bit executables for Windows 3.1 to run on 32-bit versions of Windows NT, 2000, XP, Server 2003, Vista, Server 2008, 7, and 8 [7]. Windows operating systems that are 64-bit, including Windows XP Professional x64 and Server 2012, and the 64-bit versions of Server 2003, Vista, Server 2008, 7, and 8 do not have the capability to run 16-bit applications without the use of an emulator or virtual machine. The 64-bit Windows operating systems include a subsystem, "WoW64" (Windows 32-bit on Windows 64-bit) [8] that will allow some 32-bit applications to be run on 64-bit systems. Additionally, the Windows operating system has also supported a feature called "compatibility mode" intended to allow a user to run executable files originally built for earlier versions of Windows [9], going back as far as offering a compatibility mode setting for 32-bit executables built for Windows 95. While this feature has been included in various Windows operating systems for the last decade [10, 11, 12], there is no guarantee that Microsoft will continue to include it in future versions of the operating system. Similarly, there is no guarantee that future Windows operating systems will include the compatibility layer necessary to allow users to run legacy code and older executable files. The project team found that in some cases, even if some of the executable files ran on a current system, the work performed and rendered better when using an emulator or virtual machine.

It is worthwhile to note that PC-compatible executable files may also run under Wine [13] on a Linux environment. Here it is preferable to configure the Wine environment so that it matches the Windows version referenced in the work's stated system requirements or was contemporary with the work's release year. Within Wine, there is the option to "install" one or more executables to simulate a Windows environment (with a specific organization of files consistent with a particular Windows installation). The project team did find that Wine was most reliable at running executable files that needed few (or no) additional required programs to render properly.

On either platform, if a work has required programs it needs to run, for example, QuickTime or a browser plugin, access can get considerably more complicated. As the number of required programs increases, so does the likelihood that shared packages and libraries needed by required programs may be in conflict [14, 15, 16], a condition known informally as "dependency hell." It is important to note that these complexities may very well have also existed for the work as it ran in its intended software and hardware environments.

Restoration potential

The restoration potential for works with executable files is variable. Here, it can be helpful to distinguish between the executable files that are created by the artist and files that are auxiliary to the work, such as third-party plugins or other software.

When considering strategies for restoring an artist-created executable file, if source code is available for the work, it might be possible to recompile the source code on a current system. If documentation is available, this can provide additional help in this process. Re-compiling on a current system may require revisions to the source code. If none of the source code is available, this will not be possible.

For auxiliary files, such as third-party plugins or other software, there are some additional considerations. It may not be possible to install the preferred software (and version) on a current system, because the installation software is no longer available. Even if it is possible to obtain the specific installer file for the required software—if, for instance, the artist included it with the work—it may not be possible to install these executable files on a current system. The newer versions of the required software may or may not be compatible with the files in the work.

References

[1] *Executable definition by The Linux Information Project (LINFO).* URL: http://www.linfo.org/executable.html.

[2] *Motorola 68000 series - Wikipedia, the free encyclopedia.* URL: http://en.wikipedia.org/w/index.php?title=Motorola_68000_series&oldid=606048541.

[3] *PowerPC - Wikipedia, the free encyclopedia.* URL: http://en.wikipedia.org/w/index.php?title=PowerPC&oldid=625091139.

[4] *Classic Environment - Wikipedia, the free encyclopedia.* URL: http://en.wikipedia.org/w/index.php?title=Classic_Environment&oldid=627118741.

[5] *Mac OS X: What is a Classic Application?* URL: https://support.apple.com/kb/TA20902?locale=en_US.

[6] *Rosetta (software) - Wikipedia, the free encyclopedia.* URL: http://en.wikipedia.org/w/index.php?title=Rosetta_%28software%29&oldid=615449686.

[7] *Windows on Windows - Wikipedia, the free encyclopedia.* URL: http://en.wikipedia.org/w/index.php?title=Windows_on_Windows&oldid=625220312.

[8] *WoW64 - Wikipedia, the free encyclopedia.* URL: http://en.wikipedia.org/w/index.php?title=WoW64&oldid=627942315.

[9] *Make older programs compatible with this version of Windows - Windows Help.* URL: http://windows.microsoft.com/en-us/windows-8/older-programs-compatible-version-windows.

[10] *How to use Windows Program Compatibility mode in Windows XP.* URL: http://support.microsoft.com/en-us/kb/292533.

[11] *How To Enable Application Compatibility-Mode Technology in Windows 2000 SP2 and SP3.* URL: http://support.microsoft.com/en-us/kb/279792.

[12] *What is program compatibility? - Windows Help.* URL: http://windows.microsoft.com/en-us/windows/what-is-program-compatibility#1TC=windows-7.

[13] *WineHQ - Run Windows applications on Linux, BSD, Solaris and Mac OS X.* URL: https://www.winehq.org/.

[14] *Dependency hell - Wikipedia, the free encyclopedia.* URL: http://en.wikipedia.org/w/index.php?title=Dependency_hell&oldid=625304085.

[15] *DLL Hell - Wikipedia, the free encyclopedia*. URL: http://en.wikipedia.org/w/index.php?title=DLL_Hell&oldid=614150870.

[16] *Extension conflict - Wikipedia, the free encyclopedia*. URL: http://en.wikipedia.org/w/index.php?title=Extension_conflict&oldid=501463436.

E — Macromedia and Related Executables Parent: Executables in Works

This classification will outline considerations for works that consist of executable and auxiliary files made using the Macromedia Director software. The PREMIS identifier used is "MACROMEDIA."

Short description

Macromedia Director was multimedia authoring software that could be used to develop graphical user interfaces and interactive games with embedded graphics and video [1]. With this software, it was possible to create ready-to-distribute CD-ROMs that contained all of the necessary files for users to run such works.[a] With Macromedia Director, it was possible for a user to create a CD-ROM with files that could be run on either a Macintosh or PC computer, or create a single CD-ROM that could be accessed and run on both platforms [2].

Implications for access

Works built using Macromedia Director often consist of executable files. As such, access depends largely on the considerations outlined for Executables in Works (See Section D).

Additionally, Macromedia-based works may contain embedded audiovisual files. It may be necessary to install additional third-party applications or programs for these embedded files to render properly within a work. Works that depend on a specific version of an application or program may have been included that on the CD-ROM, but this may not always be the case. Some works' documentation includes a list of all necessary additional third-party applications or programs, while others may not.

[a]In newer works, artists also used Macromedia Flash to create Flash and Shockwave files. The properties of those works are often similar to those created by Macromedia Director, with standalone executable files and additional auxiliary files.

It is helpful to note that the project team found that, often, a Macromedia-based work required some version of QuickTime [3] to be installed on the system. Some works' system requirements specified a particular version of QuickTime, while other works did not. Further, some works' system requirements may not have explicitly specified that QuickTime was a requirement, but some components of the work—such as sound or video—may not render properly without QuickTime installed on the system.

Restoration potential

The restoration potential for Macromedia works is variable and depends on how the artist originally created the CD for the work. Works that contain only the executable and no other auxiliary files are most likely impossible to restore or migrate to another platform. Works that contain protected Director files, which typically have an x in the extension—such as .dxr, .cxt—are likely also impossible to restore.

If the artist included any unprotected Director files on the CD—typically with an extension of either .dir or .cst—it may be possible to restore the work, though it will most likely be an extremely challenging process. It is important to note here that Macromedia Director gave artists multiple ways to protect the underlying component files that comprise the work. The difficulty in restoring a way will also depend on how motivated an artist was to keep the components of the work obscured from the user [2].

From the project team's research, restoring or migrating the work will most likely require consultation with the artist (to ensure all of the necessary files are present), a copy of Macromedia Director contemporaneous with the one used to create the original artwork (and a system—virtual or legacy hardware–to run it on), knowledge of how to use the version of Macromedia, and a working understanding of the artwork's underlying structure. The project team did find that there were some cases where an artist had already migrated a Macromedia Director-built work to another format, such as re-building the executable file for a newer platform, or generating a Flash or Shockwave version that could be run in a web browser, rather than using an system-dependent executable file. It will be worthwhile to do a preliminary investigation to determine whether a work has been migrated already by the artist when considering restoration of Macromedia-built works.

References

[1] *Macromedia Director - Wikipedia, the free encyclopedia.* URL: http://en.wikipedia.org/w/index.php?title=Macromedia_Director&oldid=29951992.

[2] *Mastering Macromedia Director 6 Chuck Henderson. CUL Catalog Record Brief View.* URL: https://catalog.library.cornell.edu/cgi-bin/Pwebrecon.cgi?BBID=6172854&DB=local.

[3] *QuickTime - Wikipedia, the free encyclopedia.* URL: http://en.wikipedia.org/w/index.php?title=QuickTime&oldid=629403146#QuickTime_3.x.

F — HFS File System

This classification will outline considerations when a work contains an HFS file system. The PREMIS identifier used is "HFS."

Short description

Hierarchical File System (HFS) is a proprietary file system developed by Apple designed for use on Macintosh computers. It was originally developed in 1985 and was eventually superseded by HFS+, which was introduced in 1998 when Apple released the Mac OS 8.1 operating system. As of 2009, with the release of Mac OS X Snow Leopard (10.6), there is no longer support for writing to an HFS-formatted volume [1]. While HFS is a proprietary format, the specification is still online [2] as of this writing.

One feature of the HFS filesystem is the existence of a data fork and resource fork for files. Data forks store "unstructured data" and resource forks store "structured data… information in a specific form, containing details such as icon bitmaps, the shapes of windows, definitions of menus and their contents, and application code" [3]. All files on an HFS-formatted volume may have a resource fork, but it is not required.

Implications for access

While HFS is technically a deprecated filesystem, it is still possible to mount an HFS-formatted disk image on certain current systems. If the work contains executable files, it will only be possible to run these executable files on their supported platform(s), however. As indicated in the classification document Executables in Works (Section D),[a] this often means access to the work must be through an emulator or virtual machine.

[a]In particular, the section describing Macintosh-compatible executables.

If the work does not contain executable files, access *may* be possible on a current system. The information in a file's resource fork may or may not be critical to access the file: therefore, there may be loss of detail or data when accessing a file that originated in an HFS filesystem on a non-HFS filesystem. Viewing the work in an appropriate emulator or virtual machine may be the best way of determining whether there is loss of significant detail when viewing the work on a current system.

Restoration potential

The restoration potential for HFS-formatted volumes is largely dependent on whether there are platform-specific files (i.e., executable files) as part of the work.

One of the most common issues with transferring files from an HFS volume to another system is ensuring that both forks are preserved in the file transfer. The `hfsutils` tools suite includes `hcopy`, which will properly transfer both forks for further inspection and evaluation.

HFS-formatted volumes can be converted into HFS+ volumes, but if it is not possible to run the actual files on the target platform (for example, a PowerPC executable will not run on a contemporary Intel-based Apple hardware), transfer to a different file system will not allow one to access the work on a current system.

Some system-independent files may have no (or non-essential) data in their resource forks. These can be transferred to an HFS+ or non-Apple file system for possible restoration and/or access. If the files can run and/or are viewable in a different system, the project team strongly suggests viewing the work in an emulator, virtual machine, or legacy hardware to identify what attributes of the work might have been affected in the transfer.

It is worth noting that artworks that only have HFS file systems were designed for specific Apple computers. Conversion to a newer system may negatively alter the aesthetic experience of the work, including the aesthetic experience of viewing the contents of the disc on a Macintosh computer.

References

[1] *Hierarchical File System - Wikipedia, the free encyclopedia*. URL: `http://en.wikipedia.org/w/index.php?title=Hierarchical_File_System&oldid=594424332`.

[2] *Data Organization on Volumes (IM: F)*. URL: `http://web.archive.org/web/20130815052517/http://developer.apple.com/legacy/library/documentation/mac/Files/Files-99.html`.

[3] *Resource fork - Wikipedia, the free encyclopedia*. URL: `http://en.wikipedia.org/w/index.php?title=Resource_fork&oldid=615966339`.