SAVING STAN:

Preserving the Digital Artwork of Joseph Stanislaus Ostoja-Kotkowski

Taryn Ellis
State Library of South Australia,
Australia
taryn.ellis@sa.gov.au

Abstract — It is not unusual to find at-risk obsolete carriers in archival collections, but these 3½ inch ADFS-formatted floppy disks hold original digital artworks from the late career of a pioneering Australian multimedia artist. The graphics files, created on an Acorn Archimedes in the late 80s and early 90s, had not been seen for more than 25 years, and the difficult process of preserving and providing access to these artworks, and their associated software, highlights the fragility of the material from this era. The case study presented here discusses how the State Library of South Australia combined open-source and community software to automate the extraction and migration of obsolete content from these disks while capturing filesystem and other metadata—and discovered that emulation is not always the simplest solution.

Keywords — automation, migration, RISC OS, born digital, digital artwork

Conference Topics — From Theory to Practice

I. INTRODUCTION

The State Library of South Australia (SLSA) holds an extensive collection of material relating to the life and practice of Polish-Australian artist Joseph Stanisłaus Ostoja-Kotkowski, 1922-1994. The importance of this archive was recognized in 2008 when it was inscribed in the Australian Register of the UNESCO Memory of the World [1]. Among the 29 linear meters of material that makes up this collection are 940 3½ inch floppy disks containing original digital artworks, and related software, created in the late 1980s and early 1990s on an Acorn Archimedes computer.

The purpose of the project described in this paper was to preserve the data stored on these superseded, at-risk magnetic media, and to make that material visible via contemporary operating systems. Alongside this, the metadata captured during these processes facilitated the creation of thousands of catalog records, allowing the public to discover this collection and to access it online. Through our participation in the Archiving Australian Media Arts (AAMA) research project [2], we hope to join other cultural institutions in raising public awareness of, and increasing access to, Australia's born-digital cultural heritage.

Processing the 940 floppy disks was undertaken in three stages: (1) disks that had previously been assigned archival numbers, (2) unassigned disks, and (3) disks that were initially unable to be imaged. These stages roughly correlated with the disks' contents, with the first group containing mostly image files in an obsolete format; the second, a mix of software and images; and the third, disks that had been formatted differently, or that were of a different type. This last set also contains mostly image files, but in more common or contemporary formats.

II. STAGE ONE

A. *A Slow Start*

The 3½ inch floppy disks at the center of this project arrived at the library upon the artist's death in the mid 1990s. Their presence was noted in the catalog, and some were assigned accession numbers, but relatively little is now known about how they were processed.

An assessment undertaken in the late 2010s identified these as primarily double-density ADFS-formatted disks. A floppy disk drive and disk-imaging program [3] was purchased along with a commercial Acorn Computer emulator [4]. The project was then revived in the context of SLSA joining AAMA in 2019-2020. The first 20 disks in the series were imaged and examined. From this sample it was confirmed that the subset of disks that had previously been assigned accession numbers predominantly contained digital artworks in the form of sprite files—a graphics file with extension 'ff9' that is native to RISC OS, and not recognized by contemporary software. Like the sprite files used in video games, these can contain one or more images, known as sprites [5]. Before work on the project could begin, Covid-19 arrived and everything stopped… and then started again.

The original project plan involved imaging each disk, mounting the image in the emulator and migrating each file individually to the PNG format via the RISC OS image-editing program, *Paint* [6], with renaming steps at intervals along the way. This was labor-intensive and risked accidental modification of the files in the editing software. We also observed issues with some of the migrated PNG files, which appeared strangely elongated when viewed via Windows. Further investigation uncovered that this ungainly aspect ratio results when contemporary software fails to correctly handle the rectangular pixels that comprise a subset of the sprite files. It is interesting to note that when this initial workflow was developed the organization was using Windows 7, which was able to correctly display rectangular pixels, a capability that was dropped in the development of Windows 10.

We needed to rethink our process.

B. Towards Automation

Online research revealed that academics, computer scientists, and RISC OS and Archimedes enthusiasts have been working on various projects both to extract data from RISC OS disk images and to migrate the sprite format to contemporary equivalents—to varying degrees of success. Not all of these shared tools were open-source or written in a common language like Python, and this made testing more difficult to carry out in a security-compliant manner. As well, timelines on the project were tight as resources had been allocated based on the original project plan, which we were now frantically re-working.

From the tools tested, we selected two community-generated programs—one to extract files from the disk images [7], and another to migrate the extracted sprite files to PNGs [8]**.** These programs were then incorporated into a custom Python script that controlled an automated workflow. In the event of errors—for example if the script was unable to extract files from the disk image, or if the correct number of PNG files were not created—the disk image (and any associated files) was automatically directed to a relevant 'problem file' directory for hands-on investigation. A second script was developed to manage metadata collection, taking text files generated in the first process and adding the contents to a spreadsheet. A project spreadsheet also served to capture information we could not extract programmatically, such as the disk brand or whether the disk label contained artwork. As well, we recorded if a disk failed the imaging process, and that disk was set aside.

C. Problem Solving

All was progressing well until we reached disk 161. We noticed a distinct color shift in the migrated PNG files: they were made up of pastel shades rather than the bold, saturated colors of the preceding files.

To save memory and disk space, some image files use an index or palette to centrally store color information, with each pixel pointing to this index rather than encoding detailed color information in place.

When we examined the 'pastel' sprite files using *Paint* in the emulator, we were surprised to find that the palettes for all these files were identical, but that they bore no resemblance to the colors as they were displayed on screen. When opened in the emulator, the 'pastel' files appeared to share the same color range as the earlier files.

With stabilizing this collection as our priority, we decided to continue imaging the disks while we had resources for this task, but to halt the file extraction and migration workflow while we reassessed.

This was also an opportunity to conduct a broader survey of the collection, and we used the emulator to examine disk images taken from across the 800 accessioned disks. In this way we encountered another conundrum—some of the artworks, while still sprite files sharing the ‘ff9’ extension, were not recognized by our emulator. Attempts to render them in other emulators or via community-created tools produced mixed results: the programs that were able to render or migrate the sprite files produced very oddly colored and extremely elongated images.

We now had two, overlapping problems: files with color palettes our migration workflow was unable to interpret correctly; and file variations we were unable to view, let alone migrate.

In RISC OS the pixel resolution, pixel shape and color range of the display (desktop) is known as the screen mode and can be changed by the user. Sprite files also have a mode, and can only be displayed in that mode, or one very similar. Additionally, it is possible to create custom modes for files (and displays) that are not automatically supported by the operating system [9].

Examining the pastel files supported by the emulator revealed that they were always in mode 21, and it seemed likely that the files we could not view were in a non-standard mode or modes. We needed a way to programmatically determine a file’s mode so we could identify files that were suitable for our current migration process and devise a new strategy for the others.

We examined sample files in a hex editor and compared the structure to online sprite format descriptions—a process which was complicated by the fact that there are variances between the different generations of sprite files and between our sprite files and those described. Using the byte offset of the sprite filename as an anchor point, we located the offset of the mode number. With this knowledge, we added code to our automated workflow to detect the mode and handle the file based on that, as well as to record the mode as part of our metadata collection.

By programmatically assessing each file’s mode, we revealed that our collection holds sprite files in eight different modes. Four of these are proprietary modes, and all the files in six modes have either no color palette or an unexpected one. To manage the files in custom modes, we adapted a third program [10] so that it could recognize the distinct pixel resolutions and shapes represented by these modes and automatically output correctly formed PNGs. As well, experiments revealed that by programmatically assigning a standard 256 color palette to our mode 21 'pastel' files, we were able to output PNG files that were identical to those from the emulator, and so we applied this to the custom mode sprite files as well.

To judge the success of this tactic, we compared the PNGs migrated from the custom mode sprites to the thousands of as-yet undigitized transparencies that Ostoja-Kotkowski had taken of the artworks displayed on his computer screen. While mindful of the limitations of using these 'screenshots' to confirm color-accuracy, they nonetheless demonstrated that the PNGs we were producing to facilitate discoverability and access were an acceptable representation of the artworks. We added the third program to our automated workflow and resumed processing the disk images.

III. Stage Two

The second phase of the project was dedicated to preserving the 170 'unknown' disks, rumored to contain software. Initially the same process was followed: the disks were imaged, then the images processed via an automated workflow, with files and metadata extracted. For this subset we used a modified script with the sprite file migration streams removed.

We observed that scattered through this batch were more disks that solely contained artwork, and we wrote a script to find and move those disk images for reprocessing via our original workflow. We then sorted the remaining ff9 files by size to determine if any artworks had been overlooked and processed those that matched our specifications. To our surprise, we were able to identify a few images from Stage One, and began to question whether these files were, in fact, created by Ostoja-Kotkowski.

It was clear that we needed to look a little more closely at this material: we had the file lists that named all the programs and their component parts (including those curious image files), but we did not understand how the material was used or how grouped material was related.

By mounting the disks images in the emulator we made some important discoveries: (1) some of the programs were 26-bit rather than 32-bit applications, which made accessing them difficult [11]; (2) a 'duplicate' application, present on several disks, was actually a series of animated electronic letters addressed to Ostoja-Kotkowski; (3) it was confirmed that some of the sprite files we had understood to be original artworks were actually demo files included with software.

The animated letters were especially intriguing. Written by a software engineer who was tailoring applications for Ostoja-Kotkowski, they provide insight into the artist's work practices and equipment. As well, we noticed that many of the programs, even those 26-bit applications we couldn't currently open, contained 'Help' or 'Read Me' documents that listed the creator's name, company or contact details, and information about the program. We identified the RISC OS text files (extension 'fff') extracted earlier, and programmatically migrated these to PDF. Among other advantages, by decoupling this content from the emulator, we could increase the discoverability of the collection and provide broader access to this research material.

IIII. Stage Three

With the majority of the disks now imaged, we were able to turn our attention to those that had failed the imaging step or were flagged as problems during the processing workflow. Looking at the latter group, it was apparent that the disk-image file sizes were varied, while the successful disk-images were all 800 KB. Testing revealed that our disk imaging software reversed the checkboxes for error handling so that when 'Write bad sectors as 0xFF' was selected, the program instead 'Skip[ped] bad sectors'. After adjusting for this and re-imaging the affected disks, we were delighted to discover none of the disks was so badly damaged that the content could not be accessed and extracted.

The disks that could not initially be imaged fell into two categories: (1) high-density disks (with their distinctive additional hole opposite the write-protection tab); and (2) disks that were formatted for DOS. By covering the additional hole with sticky-tape, we were able to image the high-density disks as double-density ADFS disks and process as before. Given the fragility introduced by the initial writing process we were lucky this worked [12]. The DOS formatted disks were imaged as '.img' files and their content accessed via Windows.

V. Wrapping up

A project to preserve archival material such as this does not conclude when the final disk has been imaged or file migrated.

The physical disks have now been photographed to capture the annotations, doodles, and notes on their labels. An archival model has been developed to reflect the status and relationships of the files, and thousands of catalog records generated—utilizing the metadata we recorded along the way. An ingest plan has been designed to guide the movement of all this material to SLSA's cloud-based digital preservation system Preservica. At the time of writing, not all of these tasks have been completed, but progress is steady.

Although the role of emulation in the project was more limited than initially anticipated, it will become all-important in providing access to some of the software and peripherals uncovered. We will need to develop strategies: for managing the technical difficulties posed by application dependencies; for training new users in how to interact with a very different computing environment; for sharing this material beyond the single configured machine in the lab. These questions are much discussed in the Preservation community, and now that Stan's archive is safe, maybe we can take some time to look at them too.

Acknowledgement

The Archiving Australian Media Arts (AAMA) research project is funded by the Australian Research Council.

References

[1] UNESCO Australian Memory of the World Committee, "Documenting Visual Arts in Australia: Archives of Joseph Stanislaus Ostoja-Kotkowski," *The Australian Register UNESCO Memory of the World Program.* [Online]. Available: https://www.amw.org.au/sites/default/files/memory_of_the_world/documenting-visual-arts-australia/archives-joseph-stanislaus-ostoja-kotkowski.html. [Accessed: June 2023].

[2] Archiving Australian Media Arts, "Archiving Australian Media Arts: Towards a method and a national collection." [Online]. Available: https://aama.net.au/ [Accessed: June 2023].

[3] J. Watton, *Omniflop Wizard*. (Version 3.0). Sherlock Consulting Limited. [Online]. Available: http://www.shlock.co.uk/Utils/OmniFlop/OmniFlop.htm [Accessed: June 2023].

[4] G. Barnes and A. Timbrell. *Virtual RPC Adjust SA* (Version1.7.1.0). 3QD Developments Ltd. [CD-ROM]. Available: www.virtualacorn.co.uk/products/vrpcadsa.htm [Accessed: June 2023].

[5] R. Amos, "Pixel Graphics and Paint," in *Graphics on the ARM.* Swadlincote, UK: RISC World, 2007. [Online]. Available: http://www.riscos.com/support/users/grapharm/chap08.htm#L0005 [Accessed: June 2023].

[6] Acorn Computers Ltd, Cambridge, UK. *RISC OS 3.7 User Guide*. (1996). [Online]. Available: http://www.riscos.com/support/users/userguide3/book2ab/e_2.html#marker-9-47 [Accessed: June 2023].

[7] T. McLaughlin, *py3adf* (Version 0.1.2). [Online]. Available: https://github.com/jarpy/py3adf [Accessed: June 2023].

[8] S. Huber, *SpriteViewer/SpriteConverter* (Version 0.3.0). hubersn Software. [Online]. https://www.hubersn-software.com/; https://www.huber-net.de/risc_os/ Available: https://stardot.org.uk/forums/viewtopic.php?t=15801) [Accessed: June 2023].

[9] Acorn Computers Technical Publications Department, Cambridge, UK. *RISC OS 3 Programmer's Reference Manual*. (1992). [Online]. Available: http://www.riscos.com/support/developers/prm/modes.html [Accessed: June 2023].

[10] D. Boddie, *The Spritefile module: spritefile.py; spr2other.py* (2005). [Online]. Available: https://www.boddie.org.uk/david/Projects/Python/Spritefile/ [Accessed: June 2023].

[11] This relates to the development of the ARM chip, which started out with a 26-bit address space but 32-bit CPU processing. See the following for some interesting tidbits: https://encyclopedia.pub/entry/history/show/78043 ; https://lowendmac.com/2018/the-arm-story-riscy-business/ ; https://heyrick.eu/assembler/32bit.html [Accessed: June 2023].

[12] M. Brutman. "Working with Disks: An intro to floppy disks and floppy drives." BrutmanLabs.org. http://brutmanlabs.org/Diskettes/Diskette_handling.html [Accessed: June 2023].