Emulation Case Study

Contents

This page describes a case study in the use of *emulation* to preserve access to obsolete CAD data files. *Emulation* means reproducing the program execution environment for which a piece of software was originally developed, either in software or a mixture of software and hardware.

We attempt to run a simple CAD system from about 20 years ago in order to predict the experience of digital preservationists a few decades into our future. In this case, we chose a program released about 17 years ago for the Apple II, a very popular platform. We demonstrate how it was made to work on an emulated Apple IIgs system to prove it is possible, and discover the pitfalls that should be be avoided.

Target Software and Platform

We chose the target platform, the Apple II microcomputer, because it seemed to be an ideal candidate for preservation through emulation: The original Apple II was thoroughly documented in the Red Book technical reference manual, so accurate emulators *ought* to be available. It was also beloved of computer hobbyists, and it spawned an enormous market in "freeware" and "shareware" software written by those hobbyists and marketed to each other.

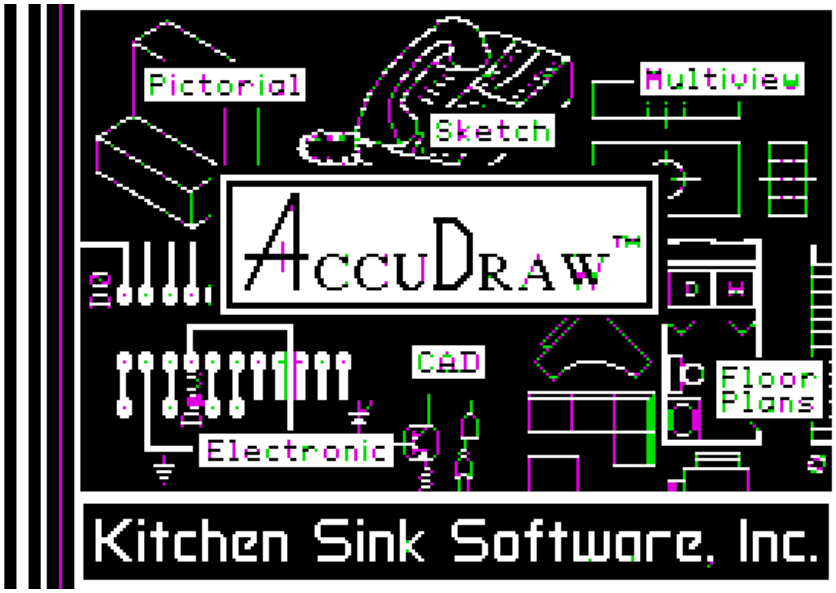
Another the Apple II is appropriate for this experiment is that it is different in every respect from the currently prevalent computing platforms: It is an 8-bit machine, while current CPU architectures are 32 to 64 bits; an its CPU family is the 6502 which is a completely different branch of evolution from the Intel x86 architecture. Some other popular 8-bit processors from that era (e.g. 8080, Z80) bear noticeable family resemblence to the x86. The greater difference makes this a harder and more useful test.

The target software was easily found through a Google search: AccuDraw was a sophisticated "paint" program that claimed to work like a CAD system. We didn't have any example drawings at hand to "recover", but we could always produce one by running the program. Also, AccuDraw has been released as free software so it can be legally downloaded and archived.

Acquiring the Software

At first glance, acquiring the software seemed trivial, but it proved more complex. The program and the manual (in easily-accessible PDF!) are readily downloaded:

1. http://www.kitchen-sink.com/a2/accudraw/assets/files/ACCUFLEP.BXY
2. http://www.kitchen-sink.com/a2/accudraw/assets/files/AccuDrawManual.pdf



Note: The thin vertical lines appear in green and purple because of a known artifact of the Apple II high-resolution graphics system.

Decoding

The Kitchen-Sink site notes its disk image is in "Shrinkit" format. We thus discover the first pitfall: accessing digital objects in obscure or obsolete compression/aggregation formats.

Fortunately the popularity of the Apple II platform and the diligence of its enthusiasts ensured there was enough information readily found on the Web to identify the "Shrinkit" and even decode it. It turned out that the AccuDraw file was actually encoded in the "BinaryII" format, the output of an archiver that bundles together a group of files (much like the current Unix utility "tar"). Since the archive contained only one file, its only contribution was to preserve the original filename (which indicates it is a Shrinkit product).

Unwrapping BinaryII

Again, we were saved by the popularity of the Apple II platform and the existence of the World Wide Web, since those facts combined to virtually ensure that the specifications of those obscure bundling and compression formats were published and accessible to us.

It was even easy to find free, open-source utilities to interpret the obscure old Apple-specific formats:

1. *unblu* to interpret BinaryII, available on http://www.fadden.com/dl-apple2/
2. NuLib is an excellent open-source utility that supports several Apple archive formats, including BinaryII and Shrinkit.

The latest version of NuLib was easily built on a current mainstream Linux platform (and is also available as a binary for mainstream Windows). It unwrapped the BinaryII file without any trouble, ...revealing a Shrinkit compressed file.

Unwrapping Shrinkit

Fortunately, NuLib is also capable of uncompressing Shrinkit files. It recovered the original 800-kilobyte floppy-disc image without any trouble. The only indication of success was that the file is the right size, however; there was no checksum, signature, or any other indication of provenance to assure us that we'd recovered the actual software.

Now What?

We had the AccuDraw software in the form of a binary file that purports to be an exact image of the contents of the original floppy disc. Since this is the way Apple II software was usually distributed, that is actually a useful medium. However, it does preclude examining it easily, since the disk image is a filesystem.

AppleCommander to the rescue

The vibrant Apple II community once again provides a solution: AppleCommander reads the major types of Apple II filesystems and transfers files to and from your "host" operating system. It lets us explore the AccuDraw disk image and provides some assurance it is viable.

Evaluating Emulators

There are dozens of Apple II emulators to be found for free downloading, and even a couple of commercial options. For the experiment we only tried free and/or open-source emulators that can themselves be preserved without any licensing issues.

Criteria

- Does it run on 32-bit x86 Linux?
- Does it run on MS Windows XP?
- Can it mount 800K ProDos 8 floppy images?
- Is source available?
- Can it legally be archived?

Apple II Emulators

Out of the many available emulators, almost all of them had at least one fatal flaw preventing its use for this application.

JappleII

JappleII appears to be vaporware. Too bad, since purports to be "a complete Apple II/II+ Emulator written in Java to be capable of running from Java Webstart".

AppleLet

AppleLet is an Apple II emulator in a Java applet. Unfortunately it is still crude and incomplete, and it does not support 800Kb floppy images.

AppleWin 1.14.0

AppleWin 1.14.0 is another Apple II simulator that only runs under Windows. However, it works under the Wine *Windows environment emulator* on Linux, which is bizarrely gratifying. Unfortunately, it also fails to read 800Kb floppies.

AppleWIN 1.3

This AppleWIN is apparently a different project from AppleWin. It does *not* get along with Wine, and even under Windows XP it would not start. Since it was last updated in 1997, we decided not to spend any more effort on it.

LinApple

LinApple is a WinApple workalike for Linux. It failed to run any code after a CPU reset, and wasn't promising enough (e.g. wrt. 800Kb disk images) to warrant further effort.

APPLEEMU

APPLEEMU is an old Windows-based emulator, that also seems to have succumbed to software rot. Couldn't get it to work even on real Windows XP.

ApplePC

ApplePC is yet another Windows-based emulator. It doesn't support 800Kb floppies.

Dapple 1.5

Dapple is currently-maintained emulator for Linux/Unix and Windows, open source too. Unfortunately the Linux port doesn't read 800Kb floppies either, and the windows executable was broken.

YAE 0.7

YAE is "Yet Another Emulator" ...that doesn't read 800Kb floppies. Do you notice a recurring theme here? It does build and work nicely on a Unix system with X11.

KEGS

KEGS 0.91 is an Apple IIgs emulator. The name stands for "Kent's Emulated GS". Since the Apple IIgs was *itself* an Apple II hardware emulator, after a fashion, I was reluctant to try a IIgs emulator at first. The IIgs was built on the 65816 processor which retains support for 6502 8-bit code, but it is *not*, strictly speaking, an Apple II. However, it is supposed to run *most* Apple II software, including the OS (such as it was).

One advantage of the IIgs emulator is that 3.5" 800Kb floppies were the mainstream media of the IIgs, and as expected, the emulator supports 800Kb

floppy images.

At last, success! KEGS had no trouble reading the AccuDraw disk image and running the software, both on Linux and Windows.

KEGS meets all the other criteria too: it is open-source, and even includes a native MacOSX application so it makes the trifecta of current mainstream platforms.

All it lacks is a copy of the IIgs ROMs, which is essential to its operation. Fortunately they are easy to find.

The ROM pitfall

The exact *ROM* (read-only memory) image is an essential component of every emulated Apple II system. Not only does it ensure the correct behavior of the system, but since many applications access undocumented and/or unintended features of the ROM directly, it also has to be the exact byte-for-byte image, not just a workalike (which is sufficient for e.g. some IBM PC BIOS emulators).

Unfortunately, the legal status of the contents of the Apple II series' ROMs adds much uncertainty and complexity to the tasks of distributing, and preserving, emulators. Apple retains the copyright on the contents of the ROM, so it is illegal to make them available for download and, presumably, archiving -- especially in a public digital repository.

Many emulator projects don't include any ROM images in their distribution and just tell users to "acquire" the ROM images for the models they wish to emulate, warning that they are copyrighted by Apple. Some even describe how you may legally possess and use a ROM image only if you own the hardware (i.e. an Apple II computer) it came from -- but this is not practical or scaleable.

Fortunately, Apple Computer Inc. does not seem to be interested in protecting its interest in the ROM images, because a simple Web search for e.g. `"apple IIgs rom image"` turns up useful results in the first page. It is an open question whether this will continue to remain the case.

File Formats Again

One final indignity of the ROM issue is the file format of the actual ROM image. Fortunately most emulators seem to require a simple binary file that is a byte-for-byte image of the contents of the ROM. It does not need any memory-location information because the location is a fact of the emulated hardware.

There is a plethora of memory-image formats from the realm of microcontrollers and hardware debugging, and a few Apple II emulators require ROM images in one of those formats. Format conversion adds yet more uncertainty and effort to an unwieldy task.

Results: You Try It

Here is an example of a preserved "CAD" document, and the executable environment in which it runs. Since AccuDraw saves a drawing as a whole directory hierarchy, it is provided as a disk image. It would be possible to extract the files and replicate the hierarchy in a Zip file, and an actual preservation project would probably do that, but for this demonstrate it would just create pointless extra work.

Materials to Download

- KEGS 0.91 software (incl. win32 and Mac executables)
- config.kegs configuration file
- Apple IIgs ROM image
- AccuDraw bootable disk image
- Data disk image with example drawing

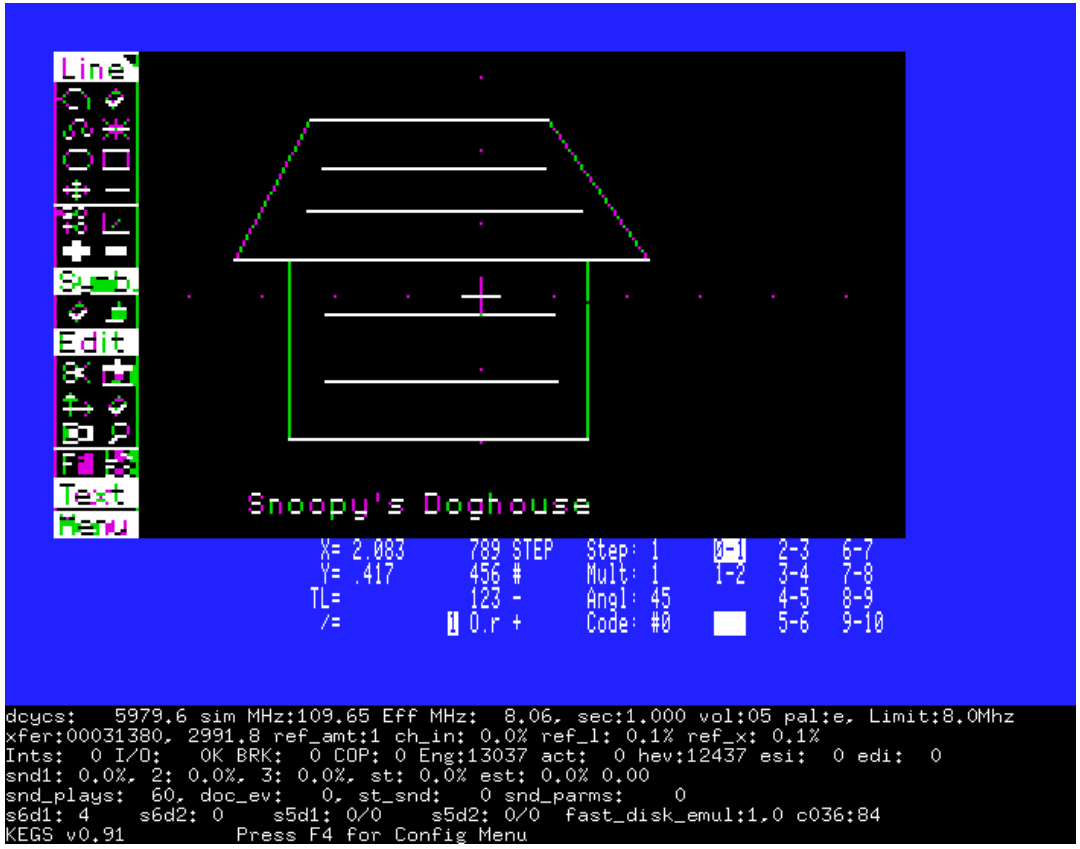
Procedure

These directions are for Linux, but Windows is very similar.

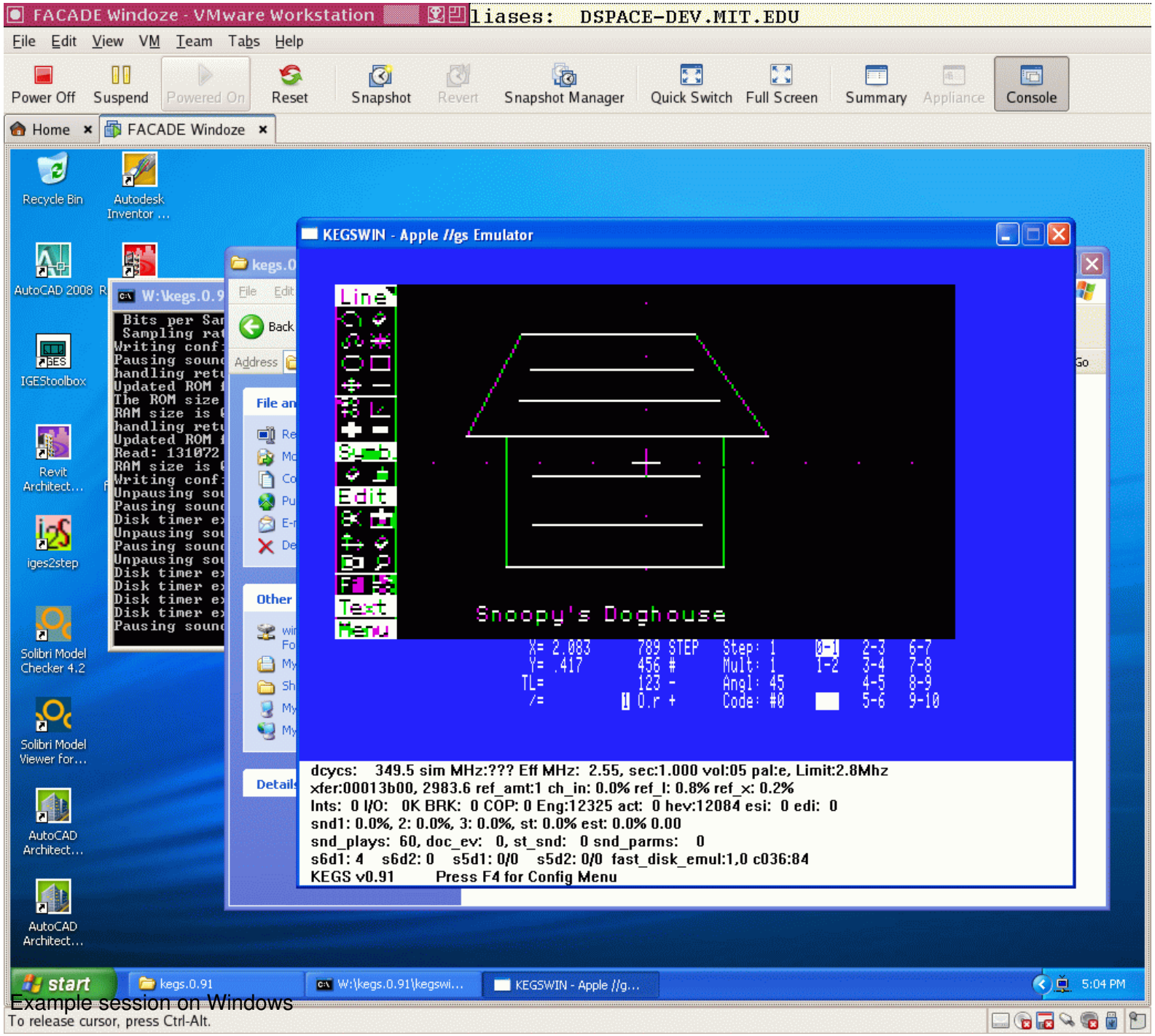
1. Download KEGS source first, unpack it with *e.g.* `tar xvfz kegs.0.91.tar.gz`
2. Download everything else, in the order given, INTO the new `kegs.0.91` subdirectory, and `cd` there.
3. Build KEGS for x86 Linux (skip for Windows or mac)
    1. `cd src`
    2. `rm vars; ln -s vars_x86linux vars`
    3. `make`
    4. `cd ..`
4. Replace config file with ours:
    - mv Config.kegs config.kegs
5. Start KEGS with the command:
    - `./xkegs -audio 0`
6. It should boot into the Kitcen Sink SAC menu.
    1. Select ACCUDRAW from the meny with cursor keys or by typing "A", hit RETURN.
    2. When it gives the Main Menu page, open file:
        1. Cursor down to `File`, hit RETURN.

2. Cursor down to `Open File`, hit RETURN.
3. Cursor UP to `Other Disk`, hit RETURN.
4. Cursor down to `Slot 6 Drive 1`, hit RETURN.
5. Cursor down to `SNOOPY2`, hit RETURN.
6. Cursor to select layer `L1`, hit RETURN.
7. You should see a crude elevation of a doghouse (sorry, Snoopy's away).
8. To quit, hit SHIFT-F6 to break out of the emulator into its debugger; type "Q" to quit.

Example session on Linux



Example session on Windows



Example session on Windows

Lessons Learned

Here are the major facts we learned from this project about digital preservation by emulation:

- The *popularity* of the target computing platform is important: More popular ones have more people working on emulators and preserving documentation and other necessary materials.
- A *thoroughly and accurately documented* platform such as the Apple II series, and even the original IBM PC, allows more accurate emulators to be built.
    - Beware of crucial elements such as the Apple II ROM images in this example which are unavailable or legally restricted, making open preservation effectively impossible.
- Avoid *obscure media*: Every platform has its *mainstream* media, such as the 140Kb 5-1/4" floppy for Apple II, and the 360Kb floppy for IBM PC.
    - Then there are less-mainstream-yet-still-vendor-standard media such as 800Kb 3.5" Apple II/IIgs floppies, or the horrifying Apple Lisa disks. These will make your data-access task harder, since fewer emulators support them.
    - The worst are third-party vendors with their own media, usually for backup or interchange, such as the many QIC (quarter-inch-cartridge) tapes. These are often a lost cause.
- Avoid *wrapper* (aggregate) and *compression* formats.
    - Even the dominant compression utility of the Apple II era, Shrinkit, will be obscure 20 years later.
    - Storage is always getting cheaper, don't use compression.
- Technical (file format) metadata is important -- especially for files removed from their chronological context.
    - Example: the ACCUDRAW.BXY disk image was misidentified as "Shrinkit", was really "BinaryII".
    - Fortunately we found Web sites documenting these formats, because they are not in the scope of e.g. PRONOM and LoC.
    - It was also fortunate that those aggregation and compression formats *were* openly documented, allowing Linux utilities to be developed. If your format is obscure, at *least* make it open.
- The Web was an essential tool. That was how we found emulators, format translators, documentation and data.
    - In another 20 years, Apple II information may be scarce and hard to find, so the Wayback Machine might be the only way to find the needed resources.
- None of the resources we relied on came with any provenance outside of the website where they were found.
    - Checksums and especially digital signatures would be helpful to identify the source and ensure integrity.

In conclusion: The goal of accessing a drawing written by an obscure and obsolete "CAD" system was achieved. The materials and documentation needed to make it possible on several current mainstream computing platforms *can* be gathered and preserved. However, one essential element -- the Apple IIgs ROM image -- *cannot* legally be archived in a publically accessible digital repository. In an actual digital preservation project, this one small obstacle could grow into a an expensive and time-consuming legal negotiation that overwhelms the resources available for preservation of the materials.

The lesson to take away is that emulation draws on resources from many different sources, and some of them may hide legal obstacles you were not expecting. The technical issues are difficult and challenging, but the legal issues may prove intractable.

- See also Public:Virtual Environments