Game Demakes for the Apple II *

Vince 'deater' Weaver
University of Maine
vince@deater.net

March 2020

Modern video games have amazing graphics and sound — but have you ever wondered if the same gameplay would have been possible on the 8-bit systems available years ago? The reimplementation of recent games for retro-systems is common enough that there is a term for it: a "demake". It can be a fun challenge to take a game and make it playable within the hardware constraints of vintage hardware. Demakes are available for a wide variety of games and platforms, targeting pretty much every imaginable 8- and 16-bit system.

I happen to have written a few demakes myself.

Kerbal Space Program

One of my favorite games is Kerbal Space Program, originally released for Mac, Windows, and Linux in 2015. In this game you use modular parts to construct rockets that launch little green people known as Kerbals into space to explore their solar system. The Kerbals are brave in the face of danger, as the launches tend to go poorly and a lot of time is spent launching rescue missions to save those stranded by previous missions.

Calculating the proper fuel load for rockets involves a lot of math, and at some point I got curious if the Apple II could handle it. I started coding in BASIC, which is a surprisingly efficient language for rapid prototyping of fully playable games. The math for spaceflight is relatively straightforward in two dimensions and the floating point support in Applesoft BASIC is sufficient (though a bit slow) to implement a fun version of the game. Many features of the real game are included: the opening splash screen and theme music, a simplified interface for building modular spaceships, and the ability to launch into orbit with a framerate of a few frames per second.

My graphics were drawn in GIMP for Linux, converted to HGR format, and BLOADed into memory. The rockets were drawn using shape tables. The code was written with a plain text editor and built using a custom BASIC tokenizer. This was then put on a disk image for testing on an emulator before running on real hardware. The code found a bug in my tokenizer due to an interesting corner case in Applesoft, where the AT and ATN tokens need to be carefully handled during parsing as they start with the same letters.

Portal

My next demake was of the game Portal (released by Valve in 2007 — not to be confused with a completely different game of the same name released by Activision for the Apple II in 1987). I thoroughly enjoyed playing Portal, despite not encountering it until long after its initial release and it being a short game. You are a test sub-

*This originally appeared in the March 2020 edition of *Juiced.GS* magazine with the title: "Behind the Scenes: deater's demakes: Doing More with Less, Through a portal into the mist of another world"

Figure 1: **All systems go:** Kerbal Space Program launches on the Apple II.

ject being run through a series of escape rooms by a mysterious AI named GLaDOS. Escaping the rooms involves puzzle-solving using a portal gun, which allows you to shoot portal openings on flat surfaces of the room that enable instant transport from one spot to another.

My Applesoft BASIC demake was inspired by the blue and orange colors prominently featured in the game closely matching colors found in the Apple II hi-res palette.

Creating a platform game in Apple II hi-res graphics is a pain; unlike with other 8-bit systems, the Apple II has no hardware sprites. This limitation can be somewhat worked around by using shape-tables. However, colored shapes are tricky: you have to be careful about drawing on odd or even columns, or choosing which of the two black background colors are used, lest your shapes end up with the wrong colors.

The demake is fully playable, with a full physics engine (although only in two dimensions). You can go through portals, jump over obstacles, avoid being detected by sentry robots, and undertake the final battle. Only the first and last level are implemented, but if you beat the game the famous ending with the song "Still Alive" is played with ASCII art and lyrics.

That song delayed the game longer than you'd imagine. I had never been quite satisfied by the sound quality available through the speaker from Applesoft, and I got side-tracked on a multi-year side-quest to make the music sound better. I eventually developed a much more advanced version of the ending song capable of playing in either Paul Lutus' two-channel Electric Duet (for the speaker) or else as 3-channel AY-3-8910 audio (for the Mockingboard).

Despite this demake's limited scope compared to the full game, it briefly made me Internet-famous, with interviews for news articles and playthroughs appearing on YouTube.

Many people who liked the Portal demake requested a faster, more complete version. I resisted this call, as it would require moving to 6502 assembly language which is orders of magnitude more complex than the quick development possible with Applesoft. In addition, a full, more complete game would quickly run into disk and RAM size constraints. Even these BASIC demakes were large enough to require special handling to load the code high in memory to avoid overlapping with the hi-res graphics locations.

Figure 2: **Welcome to Aperture Science:** The antagonist of Portal is glad to see you.

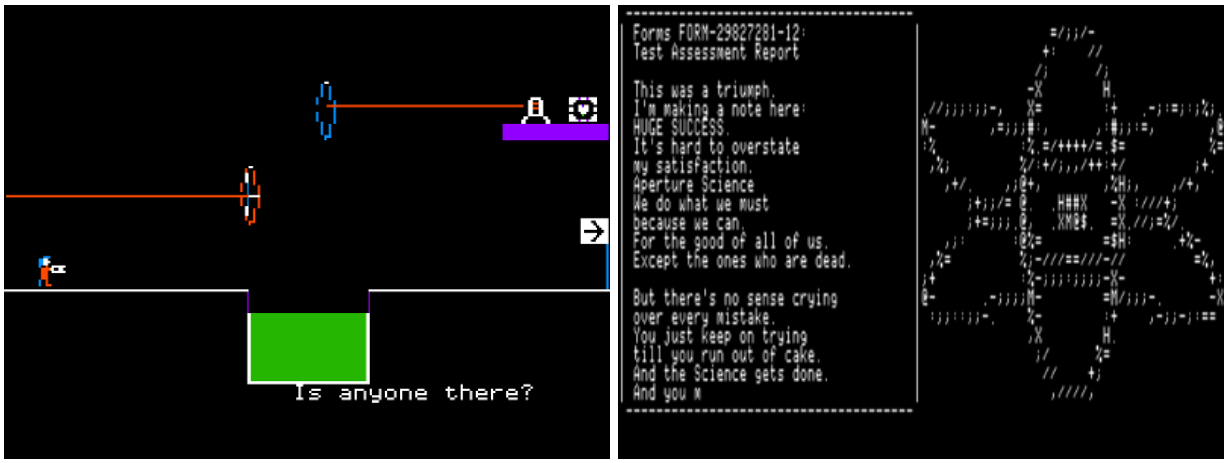

Figure 3: **This is a triumph:** GLaDOS remains as snarky as ever, even when degraded to 8-bit.

Out of This World / Another World

And yet I did end up eventually creating an assembly language demake. As with many projects, it started as a small proof of concept that rapidly spiralled out of control.

The game is Éric Chahi's 1991 classic "Another World" (released as "Out of this World" in the USA). In the game, you are a scientist who gets accidentally transported to an alien world. You have to survive on the strange new planet with a fancy laser pistol and the help of a friend you make along the way.

I started working on this after seeing Paul Nicholas's Pico-8 take on the game. (The Pico-8 is a modern fantasy console designed with similar limitations to vintage 8-bit machines). I wondered if I could make a version of the game that would run on the Apple II. You might already be thinking: wasn't this game released for the Apple IIGS? Yes, due to some amazing coding by "Burger" Becky Heineman, there was a contemporary release for that system. However, in my projects, I like to target the original Apple II from 1977, preferably fitting on on a 140k floppy and in 48k of RAM.

I decided to code the game in the oft-overlooked lo-res graphics mode, which has a resolution of 40x48 with 15 colors. This maps surprisingly well to Another World's 320x200 with 16 colors. The lo-res graphics mode has the advantage of not taking much RAM (1K for the display, as opposed to 8K for hi-res). This allows faster screen updates, smaller sprites and background images, and room for multiple off-screen pages to construct each frame before copying to the screen.

The original game was for the Commodore Amiga, handwritten in m68k assembly. The code was written in a custom interpreted language that drew single-colored "polygons" to the screen. Contemporary ports of the game reimplement that interpreter, but that probably is not possible on the original Apple II, so software sprites are used instead.

The demake graphics are hand-drawn based on screen captures gathered from the Windows 3.1 release of the original game. To fit in RAM and on disk the graphics are compressed with a simple run-length-encoded algorithm. The game tends to be dark, so the animations and sprites often include the color black. This can be a problem as traditionally black is used for transparency. The demake takes advantage of the fact that Apple II lo-res has two interchangeable grey colors and the second one is used for the transparent color.

There's a famous scene in the game where an alien talks to you (although no one can agree on what is said— one approximation is "My Tuba"). This is included in the demake at the expense of 10k of RAM holding the sound samples for the Apple II speaker. A few scenes in the game have music; when possible I have converted the original Amiga MOD music to PT3 AY-3-8910 music that can be played on a Mockingboard. This involved the creation of a PT3 music library (pt3lib) that has since been made available for other projects wanting to create music for games on the Apple II.

This demake is still under development. Currently the game is playable, with five levels (out of sixteen) fully fleshed out. In addition, the elaborate opening movie is fully implemented, as is the full end movie and credits.

Myst

My most recent demake is of the game Myst, designed by Cyan Inc. and released in 1993.

In this game, players solve puzzles while exploring a series of mysterious worlds by traveling through magical linking books. Myst was not only the best-selling computer game of the last millennium; it was also a large game, one of the

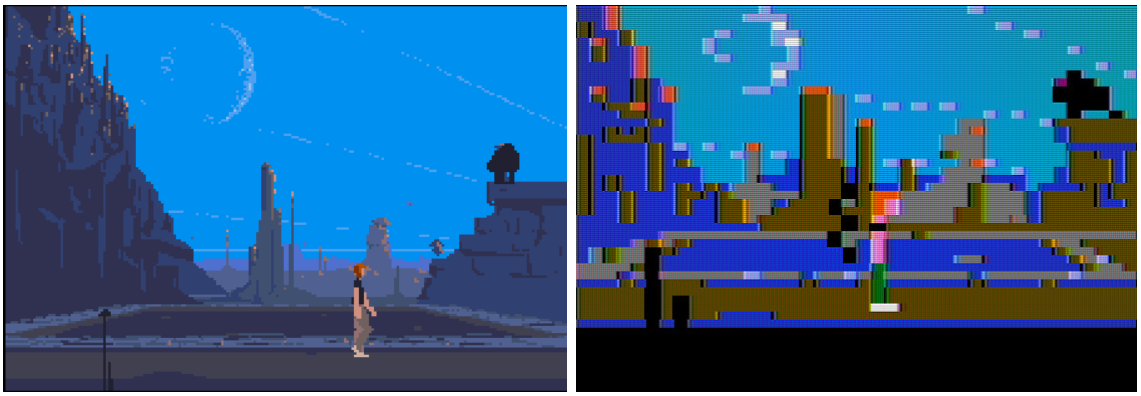

Figure 4: **Out of This World:** Éric Chahi's 1991 game was also ported to the more-capable Apple IIGS by Becky Heineman.

first to require a CD-ROM. In addition to impressive pre-rendered 3D graphics, it has atmospheric sound effects and plenty of full-motion video.

My goal is to see how much gameplay I can fit on a 48k Apple II+ with a single 140k floppy disk. This involves optimizing the size of the original game by many orders of magnitude. Fortunately, Myst's gameplay and thus the demake's code, is relatively simple: aside from the puzzles, Myst is similar to walking through a slideshow. In fact, it was originally written as a HyperCard stack! That leaves the slideshow's images taking up most of the storage. I am again using lo-res graphics, which I create by tracing over screenshots from the actual game. When compressed with the LZSA algorithm each screen averages about 512 bytes in size, which allows fitting hundreds of images on disk.

Some people, surprised that I hand-drew each image, have suggested automating the process using software. I find the low-resolution and 15-color palette makes that difficult. Myst is a dark game with lots of misty shades of grey and brown, and these do not map well to the Apple II colors. To make the game at least somewhat understandable at such low resolution involves a lot of manual colors and shape selection.

I have also received suggestions that I should be using hi-res, double hi-res, or even super hi-res graphics modes. These modes would allow the graphics to be more faithful to the original, but at a dramatically increased file size. Automatic conversion of images to a compressed hi-res graphics format are around 4kB in size each— eight times larger than their lo-res equivalents. This means instead of taking one floppy disk, the game would take eight. These larger filesizes might not be a big deal in these modern days of SD or USB storage used by devices like the CFFA3000, but I prefer to try to keep things to a single floppy.

The demake does support playing sound, even though the Apple II speaker is not know for its sound quality. My sound files are in the BTc format, a one-bit encoding designed by Roman Black for high compression levels. I use code by Oliver Schmidt that can play these files through the Apple II speaker. One iconic sound from Myst is the three-second-long noise made when traveling through a book. This 32kHz 8-bit WAV files takes 96K, which is far too big. Compressed with BTc, this file becomes 12K— although still relatively large for a machine with 48K RAM, it is nonetheless a workable size.

Development of Myst's demake is ongoing, with the plan to implement the full, main island, puzzles and all. Myst has four additional worlds to

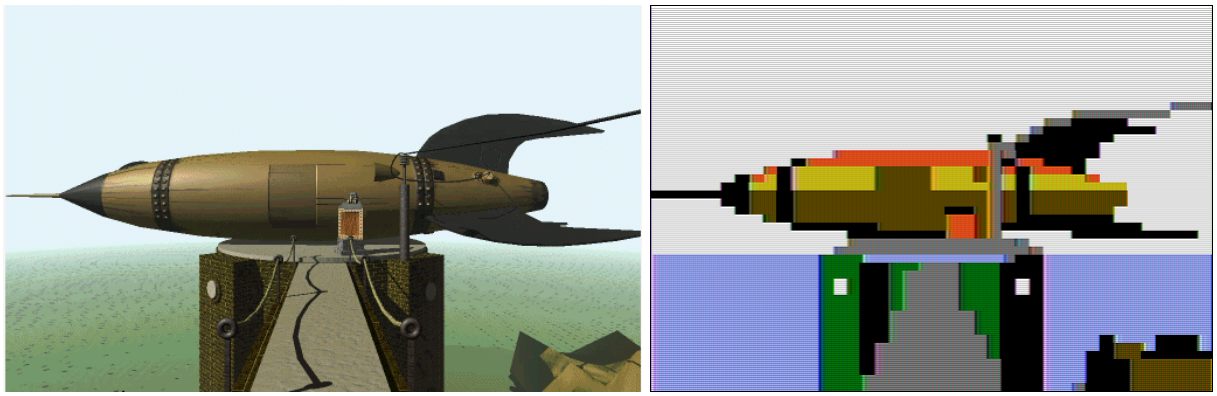

Figure 5: **Play Misty for Me:** Myst, released for the Mac in 1993, eventually sold six million copies. Within this rocket is a keyboard that must be played in the proper sequence.

visit, and it is unclear how much of that content will fit. But, much like the magical books in Myst, I may find there's more inside than I bargained for. [note from the future— eventually the full game was finished, including all puzzles, all endings, and the intro narration. This did end up requiring expansion to fit on three floppy disks]

About the Author

Vince Weaver has been programming games of varying quality for the Apple II since the mid-1980s. When he's not writing 6502 assembly language he works as an Associate Professor of Computer Engineering at the University of Maine.

Development and Source Code

I develop these games using various tools I have written for Linux. I cross-build the programs, add them to DOS 3.3 disk images, and then test using the AppleWin emulator running under the Wine compatibility layer. For 6502 code I use the ca65 assembler that comes with the cc65 compiler project.

Since the original release of this article in March 2020 many more demakes for the Apple II and Atari 2600 have been completed. All of the programs and tools described in this article, including full source code and disk images, can be found on the author's website: `http://www.deater.net/weave/vmwprod/demakes/`