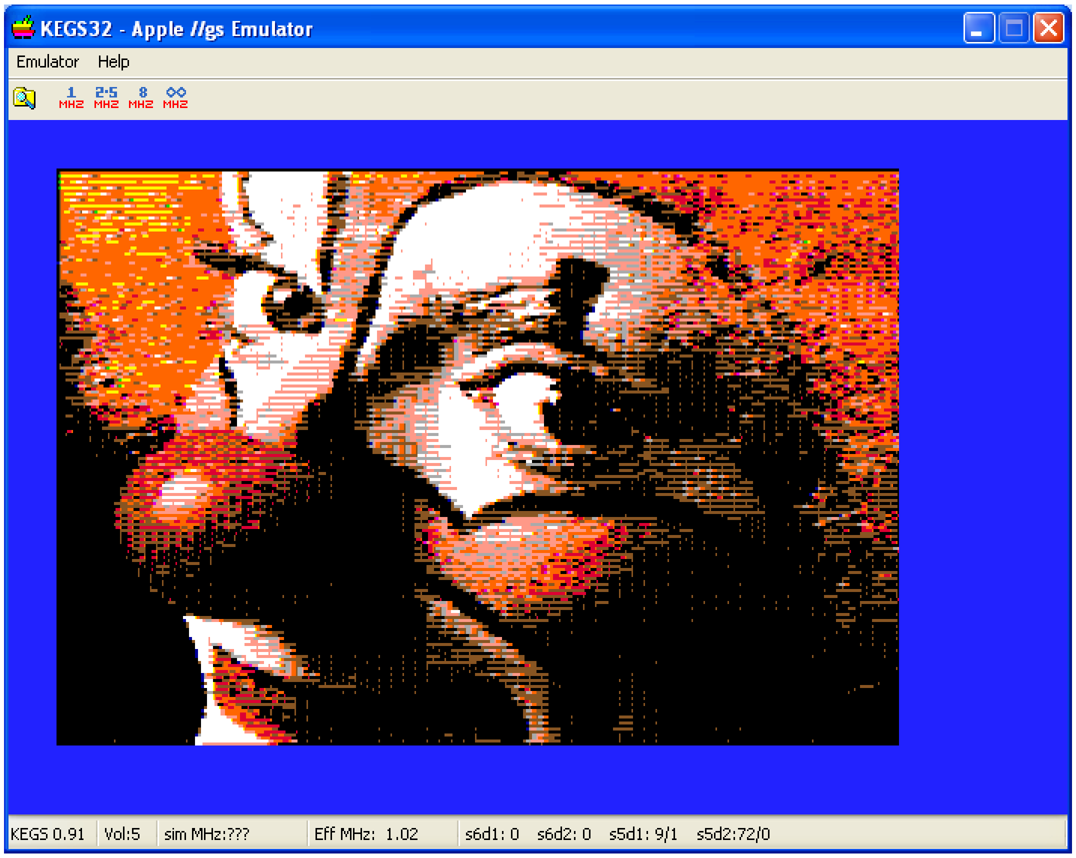
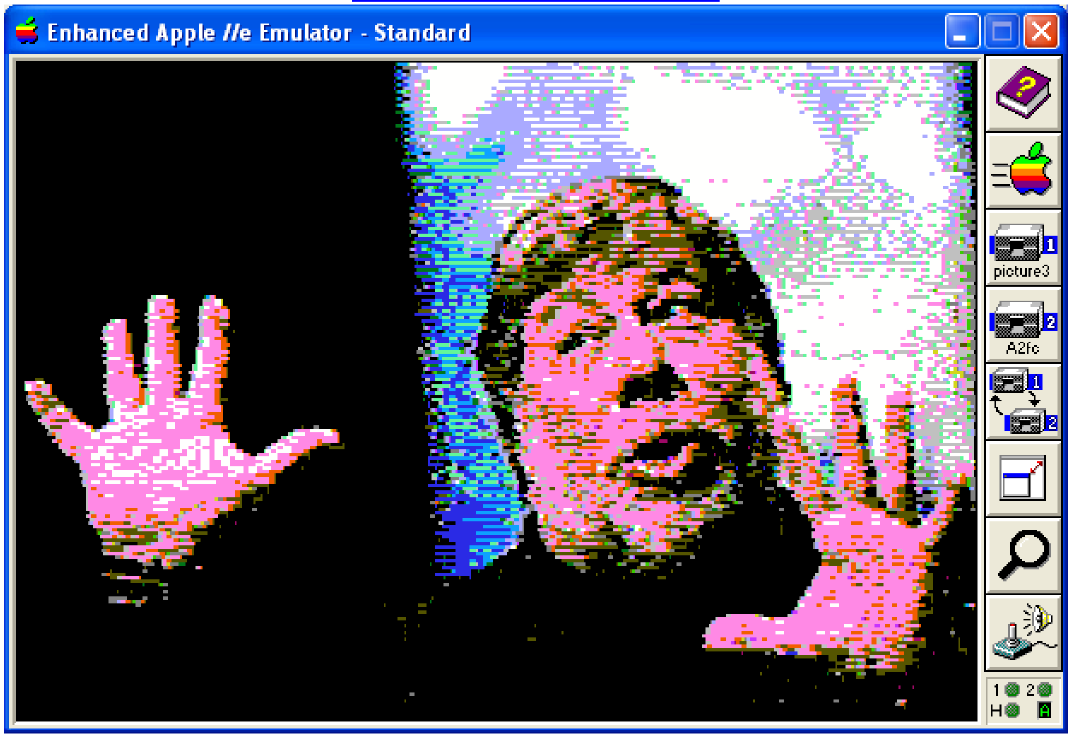
Displaying Apple II Double Hi-Res Bitmapped Graphics in a cc65 C Program



"A Visit from Saint WOZ"



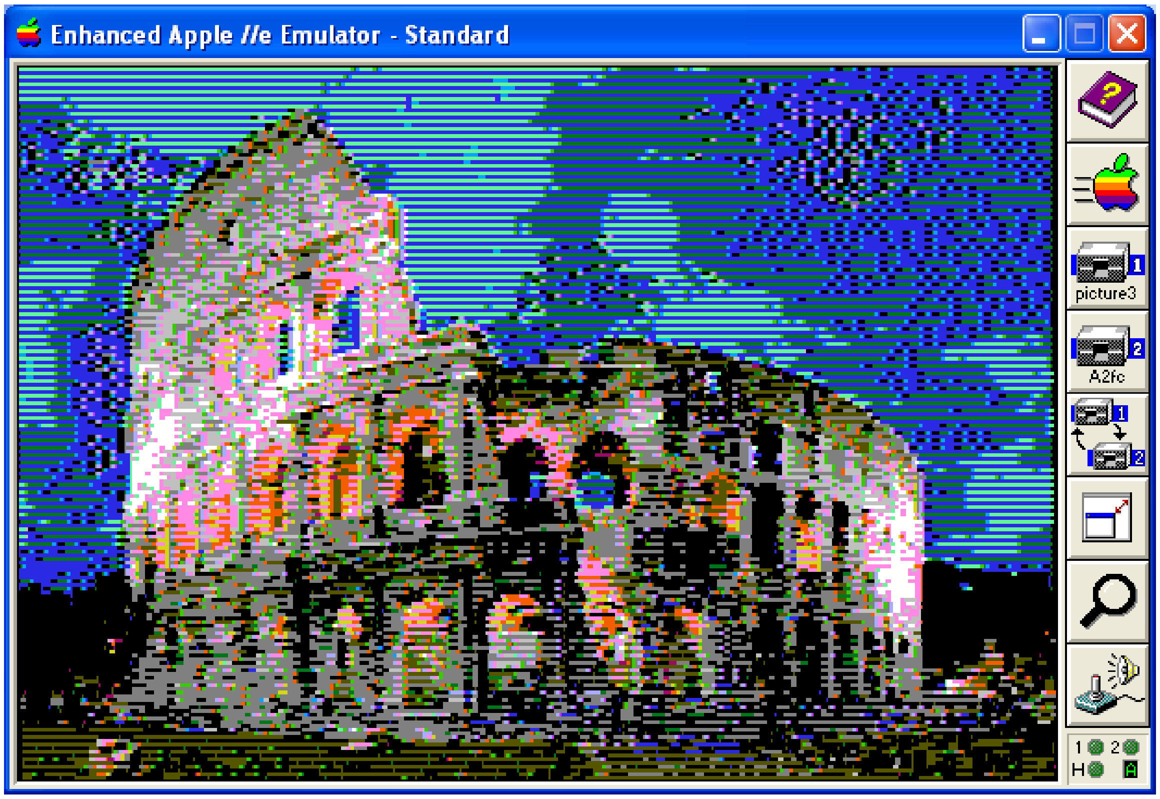
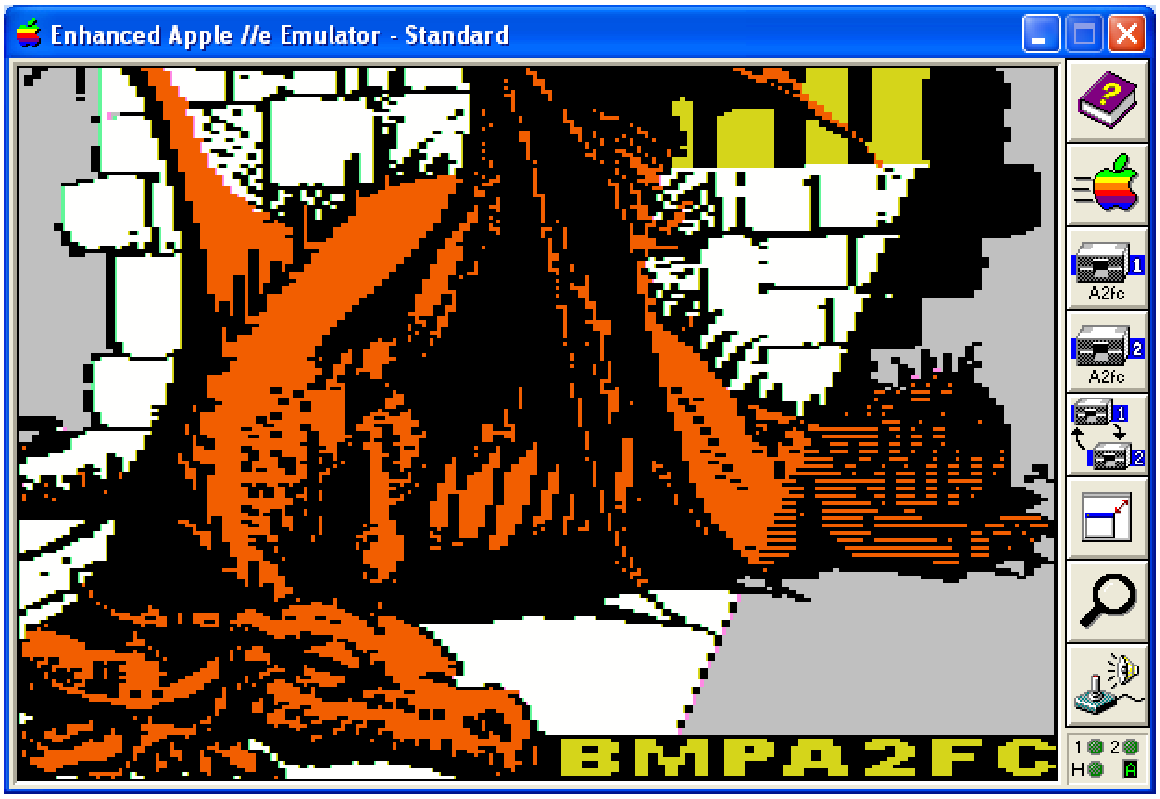
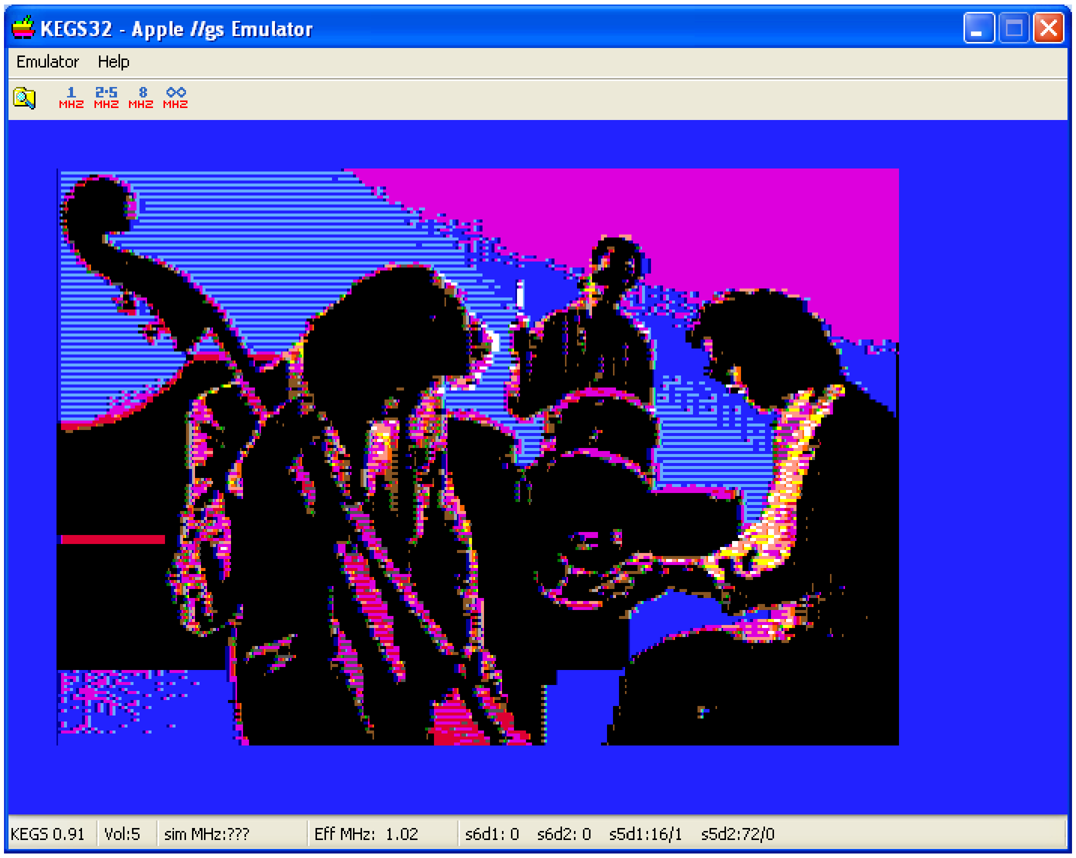
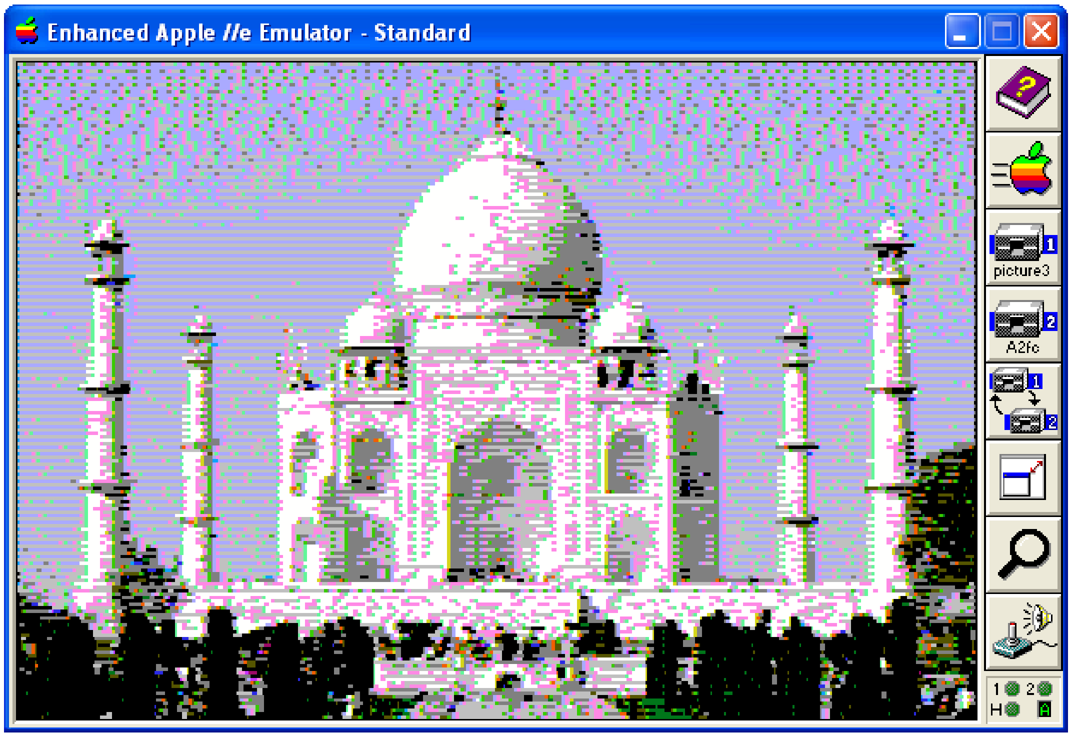


Table of Contents


Enhanced Apple //e Emulator - Standard
BMPA2FC
A2fc
A2fc

Introduction



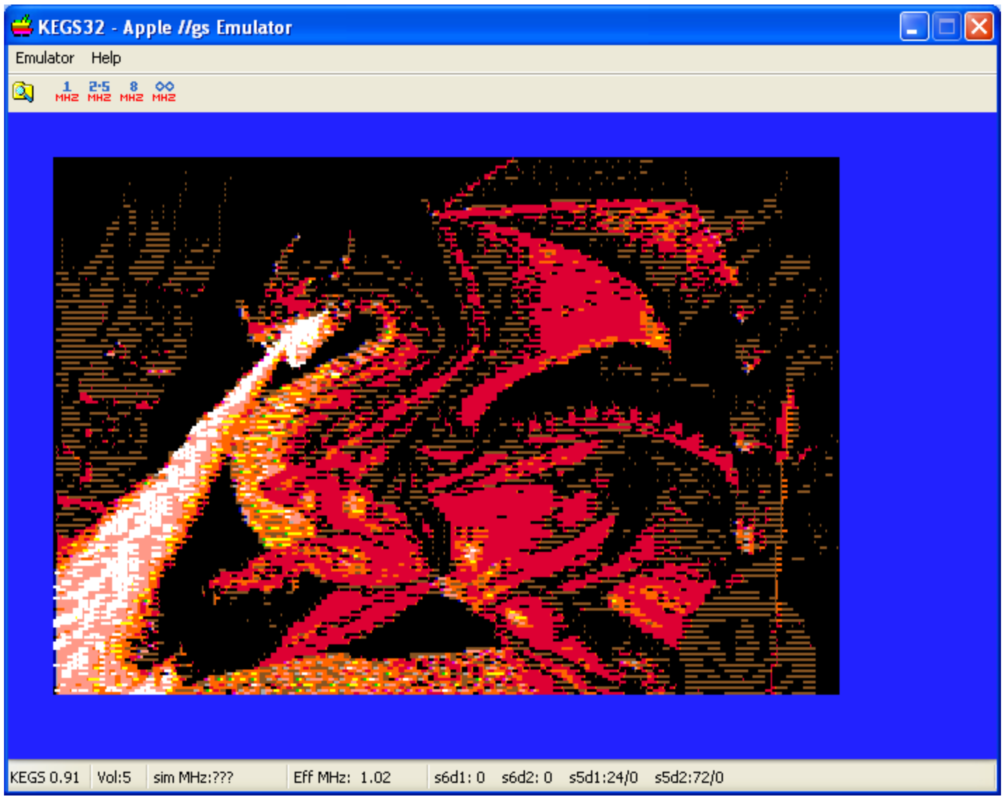


This document is about writing an Apple II Double Hi-Res (DHGR) Slideshow Program (a demo program called "dhishow") for ProDOS 8, in the C Programming Language, and about building it using the current snapshot of the cc65 compiler. Two variations of the same program are discussed; one of them uses DHGR Screen Page One at $2000 and one of them uses DHGR Screen Page Two at $4000.

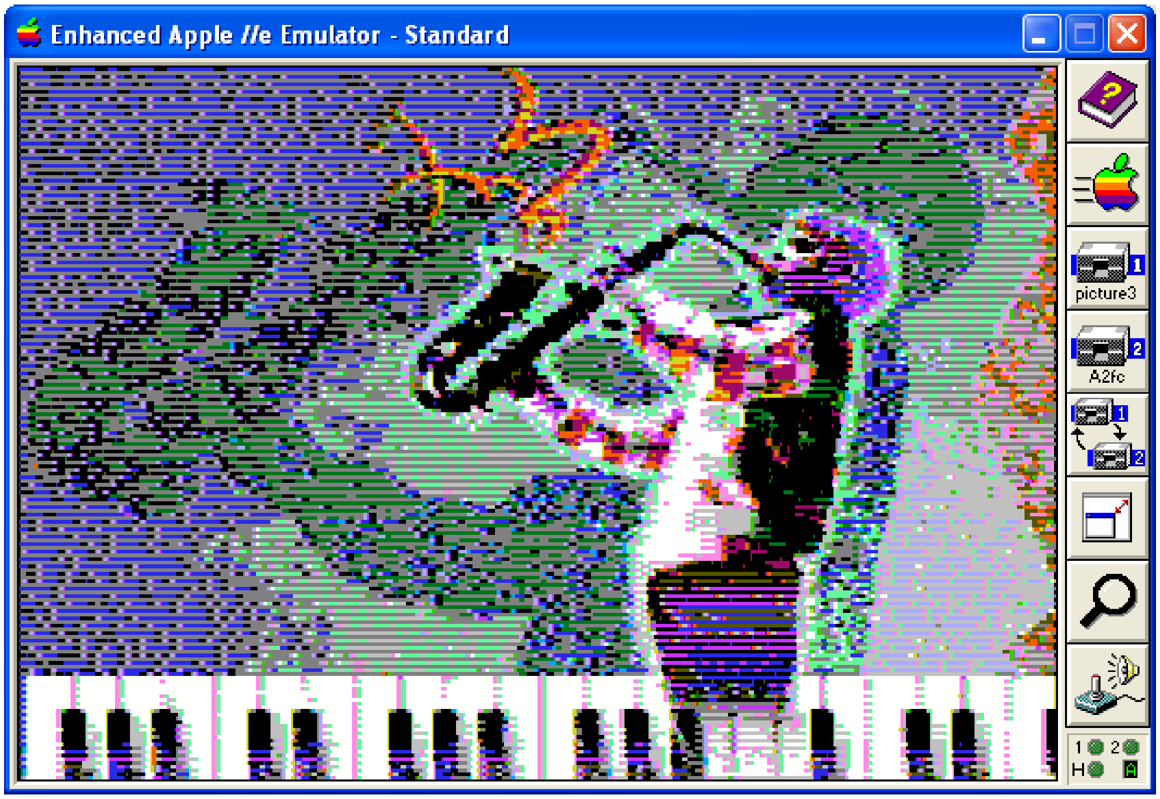
The dhishow program variation that uses DHGR Screen $2000 has considerably more memory than the other variation, because it runs above the DHGR screen and not below it; a fact that is discussed in more detail later.

The earliest version of dhishow was originally written as an Apple II DOS 3.3 demo program for the Aztec C65 Compiler in 2009, as a DHGR image loader. Between 2012 and 2013, the dlode demo was expanded into an Aztec C65 slide-show demo program for the Aztec C65 ProDOS Shell, and still used DHGR Screen $2000 and supported the same two DHGR image formats used in dhishow. In 2013 another Aztec C65 DHGR demo variation of dlode was written as a ProDOS 8 SYS program called dpshow. It used DHGR Screen $4000, and supported 2 additional DHGR image formats.

The dhishow cc65 demo program discussed in this document descended from both those Aztec C65 slideshow programs, but because cc65 uses memory differently than Aztec C65 and is also entirely different than Aztec C65 in many other ways, dhishow constitutes a full rewrite that entailed exploring the two possible dhishow variations discussed later that use Screens $2000 and $4000 respectively.


KEGS32 - Apple //gs Emulator
Emulator Help
1 MHz 2.5 MHz 8 MHz ∞ MHz
KEGS 0.91 Vol:5 sim MHz:??? Eff MHz: 1.02 s6d1: 0 s6d2: 0 s5d1:24/0 s5d2:72/0

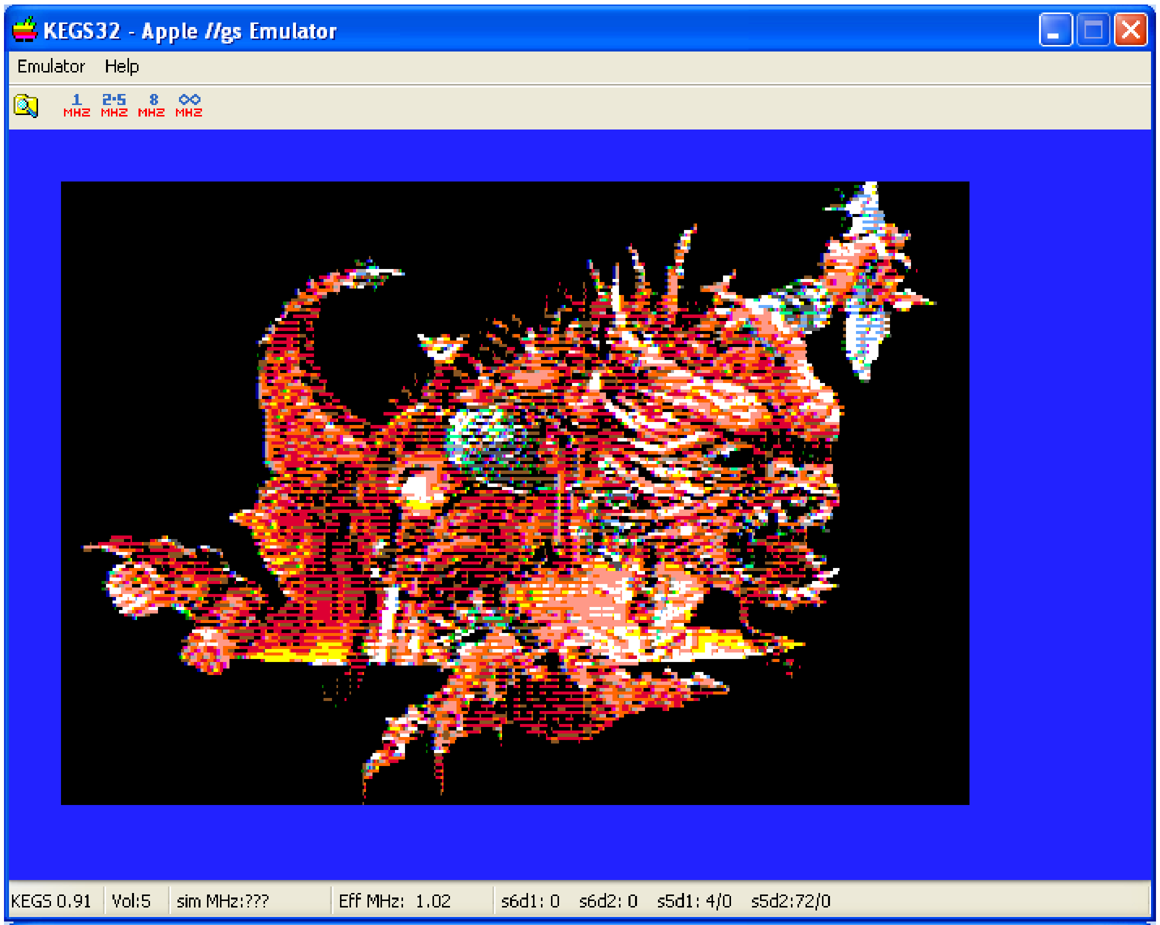
DHGR File Format Notes

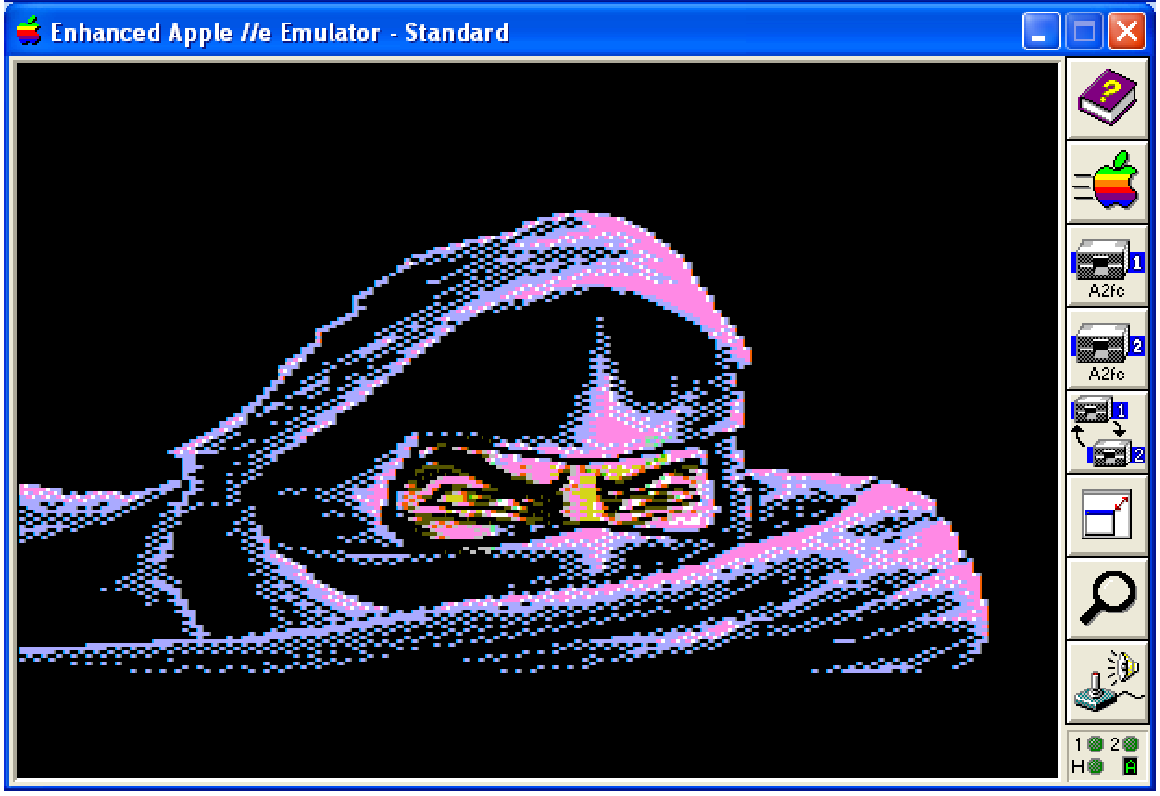


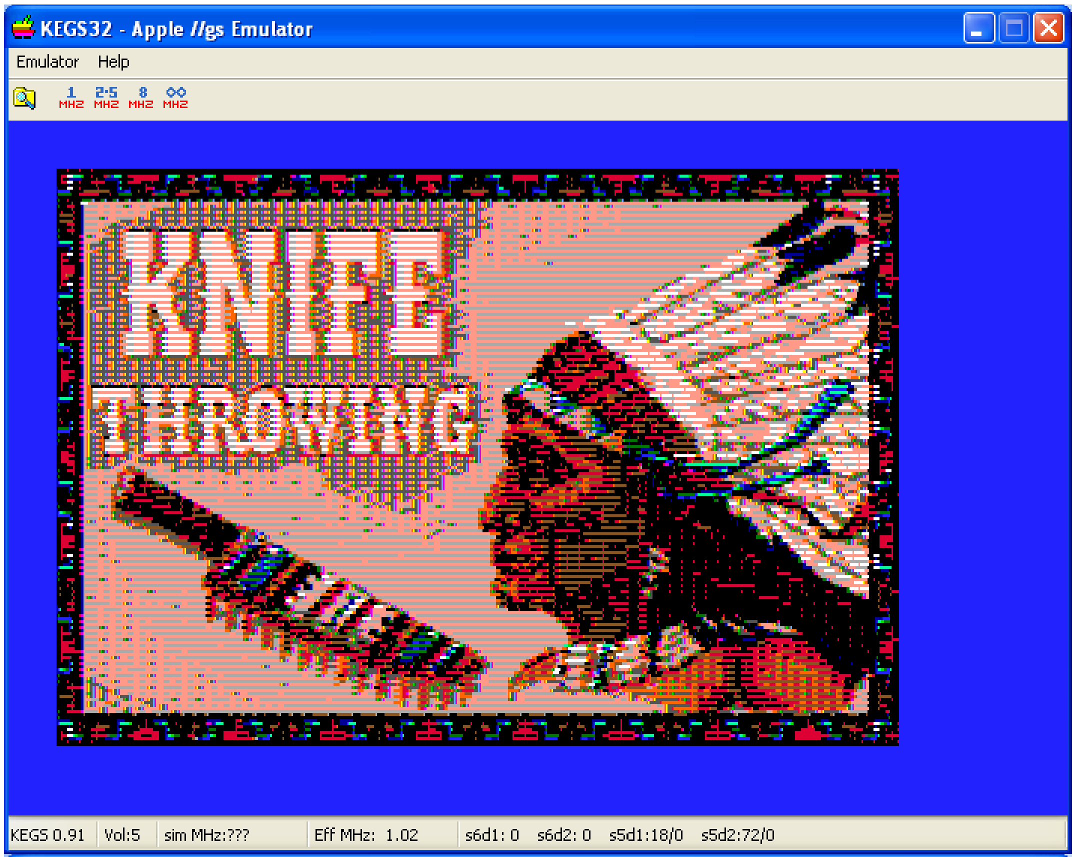
Double Hi-res Graphics Mode for the Apple II was in some ways the most capable of all the Apple II graphics modes until the Apple IIgs came along with its Super Hi-Res (SHR) graphics. Until SHR, Apple II Graphics were limited to a fixed palette. DHGR uses the same 16 colors that Double Lo-res (DLGR) uses. Like DLGR, in order to display DHGR the upper memory bank (auxiliary memory) on the Apple II must be used.

ProDOS DHGR FileType $06 Auxiliary Type $2000 or $4000

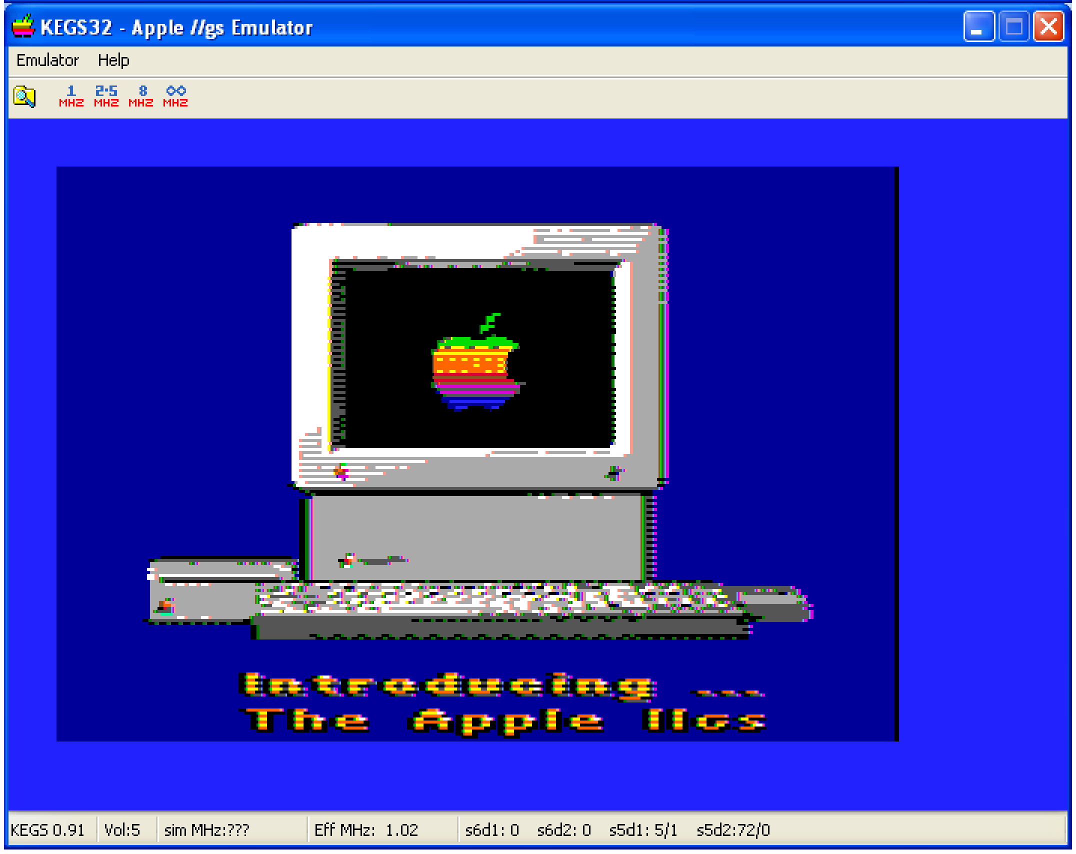
DHGR Files are stored on the Apple II as ProDOS Binary FileType $06, Auxiliary Type $2000 or $4000. In DOS 3.3 they are stored the same way, as a BLOADable Binary File Type $04 (the same as a DOS 3.3 program), with a load address of $2000 or $4000, and are appended with the load address in the first two bytes of the DOS 3.3 four byte binary header, followed by the next 2 bytes which give the length of the file itself. In ProDOS this information is stored in the filing system so DHGR files are 4 bytes longer in DOS 3.3 compared to their ProDOS equivalent.


KEGS32 - Apple //gs Emulator
Emulator Help
1 MHz 2.5 MHz 8 MHz ∞ MHz
KEGS 0.91 Vol:5 sim MHz:??? Eff MHz: 1.02 s6d1: 0 s6d2: 0 s5d1: 4/0 s5d2:72/0


Enhanced Apple //e Emulator - Standard
A2fc
A2fc



```
KEGS 0.91 | Vol:5 | sim MHz:??? | Eff MHz: 1.02 | s6d1: 0  s6d2: 0  s5d1:18/0  s5d2:72/0
```



```
KEGS 0.91 | Vol:5 | sim MHz:??? | Eff MHz: 1.02 | s6d1: 0  s6d2: 0  s5d1: 5/1  s5d2:72/0
```

DHGR Standardization - ProDOS File Type $08

In November 1988 Apple Computer decided to standardize DHGR File Types by re-assigning ProDOS File Type $08. By that time DHGR, which itself was a design after-thought on the Apple //e, had already been in use for 5 years, since the release of the Revision B Motherboard sometime around 1983-84, when DOS 3.3 still had wide use. DHGR bitmapped graphics files, with wide support in applications like Mark Simonsen's Beagle Graphics (Beagle Bros.) and David Snider's Dazzle Draw (Broderbund), had long been established as Binary Files with the Auxiliary Type being the Load Address of either DHGR Screen One or Two.

Apple Computer's after-thought in re-assigning File Type $08 also extended back to HGR (Hi-Res) files, which long predate the Apple //e and ProDOS. They also had new names for HGR and DHGR; "Limited Color" and "Full Color".

Apple's 1988 re-assignment included a description for using ProDOS FileType $08 Auxiliary Types $4000 (HGR) and $4001 (DHGR) for files compressed with PackBytes. In his CiderPress program's HGR viewer source code, Andy McFadden comments "FOT with auxtype $4000 is a compressed hi-res image. FOT with auxtype $4001 is a compressed DHR image. In practice, nobody uses these…".

Apple's re-assignment indicates that a portion of an image can be stored, but due to the way the HGR and DHGR Screens are interleaved in a non-linear manner, it is hard to understand how this might have been intended to work.

The variant of the "dhishow" demo program that runs at $4000 (described further in this document) has enough memory still available to include routines, like UnPackBytes(), to read and display most of the decipherable DHGR variants of FileType $08, but history says this is not worth the effort. So only files of ProDOS FileType $08 of 16384 bytes in length (A2FC files) are supported by this demo.

The DHGR file formats supported by this demo likely account for most of the DHGR images that you will find anywhere today. The "HomeBrew" nature of Apple II users though the time of DHGR, and to the present day, led to the AUX, BIN and A2FC formats being pretty well understood and well used. There are 3 major and over-lapping camps of Apple II users that bridge the history of the various models of the Apple II and its operating systems. There are also divisions of CP/M 80 users within those camps. Today Apple II users are also divided into 3 major and overlapping camps of modern computer users, and camps of technical users which include modern hardware users, vintage hardware users, and emulator users:

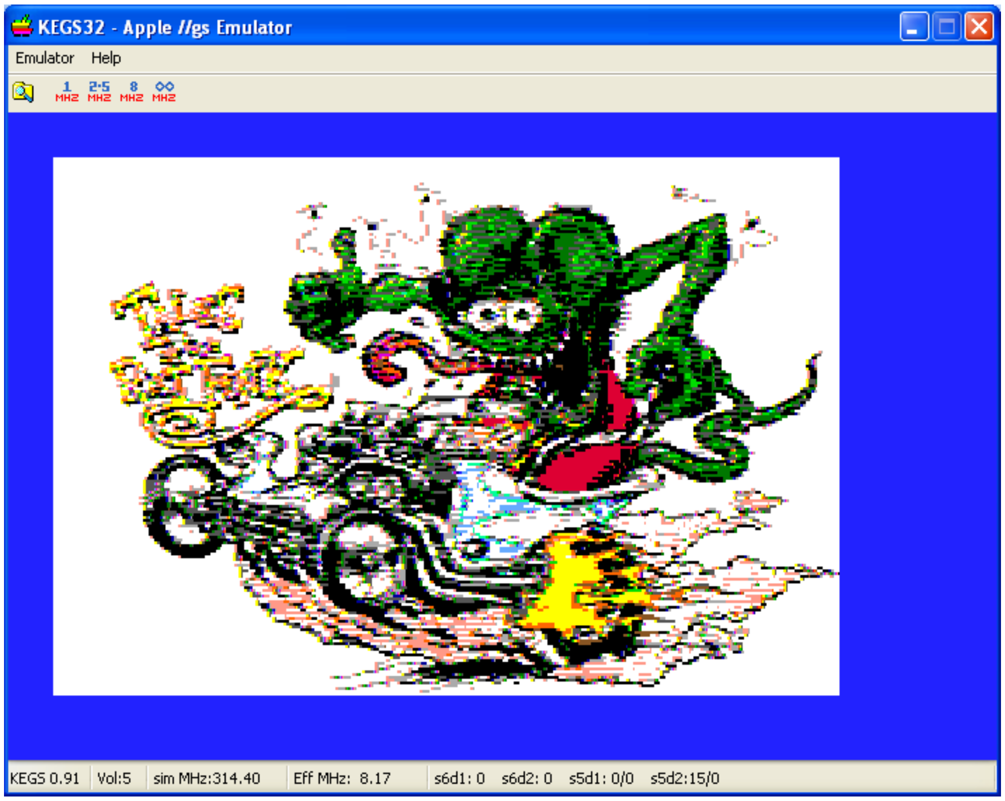
Interchangeable	**Apple DOS 3.3 etc.**	**ProDOS, SOS**	**IIgs Desktop etc.**
Interchangeable	VAX, Unix, etc.	CP/M, MS-DOS	Mac 68xxx, etc.
Interchangeable	Linux, Raspberry Pi	OS2, Windows, etc.	Mac OSX

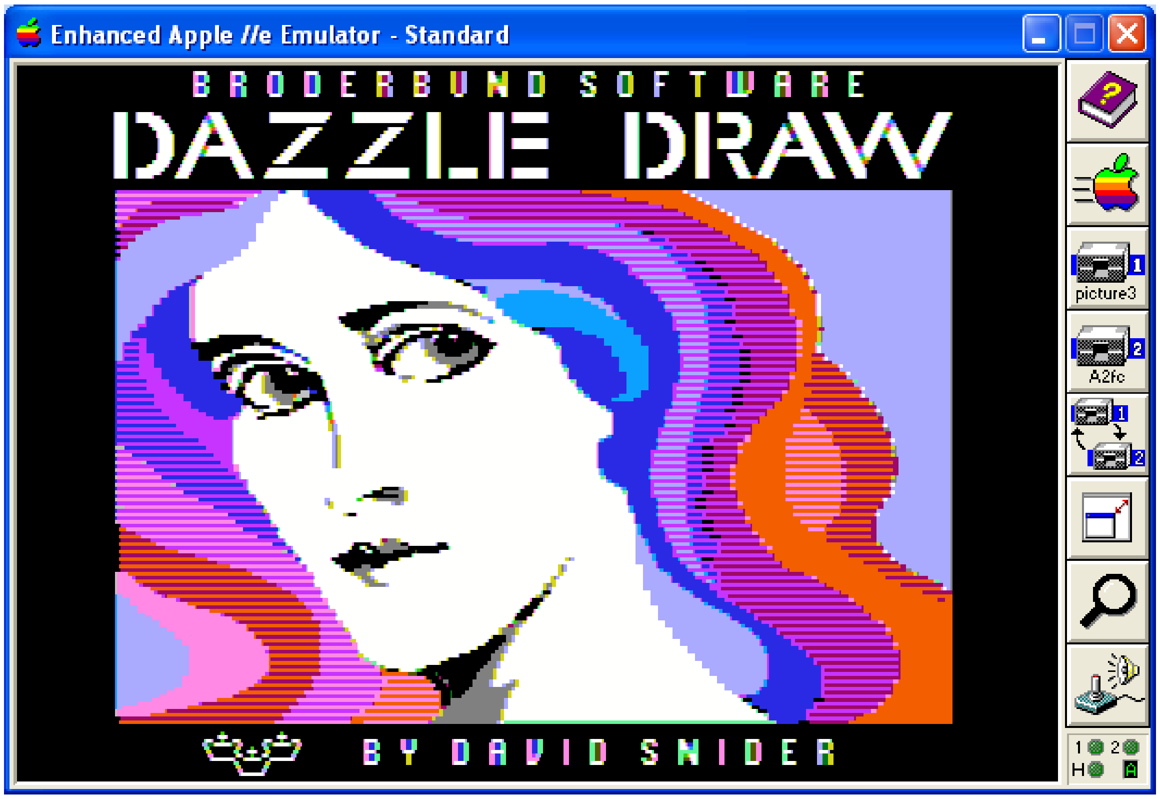
In order to better understand why Apple II users failed to embrace this then-new use of ProDOS FileType $08 for HGR and DHGR files, Steven Weyrich's book is a good place to start:

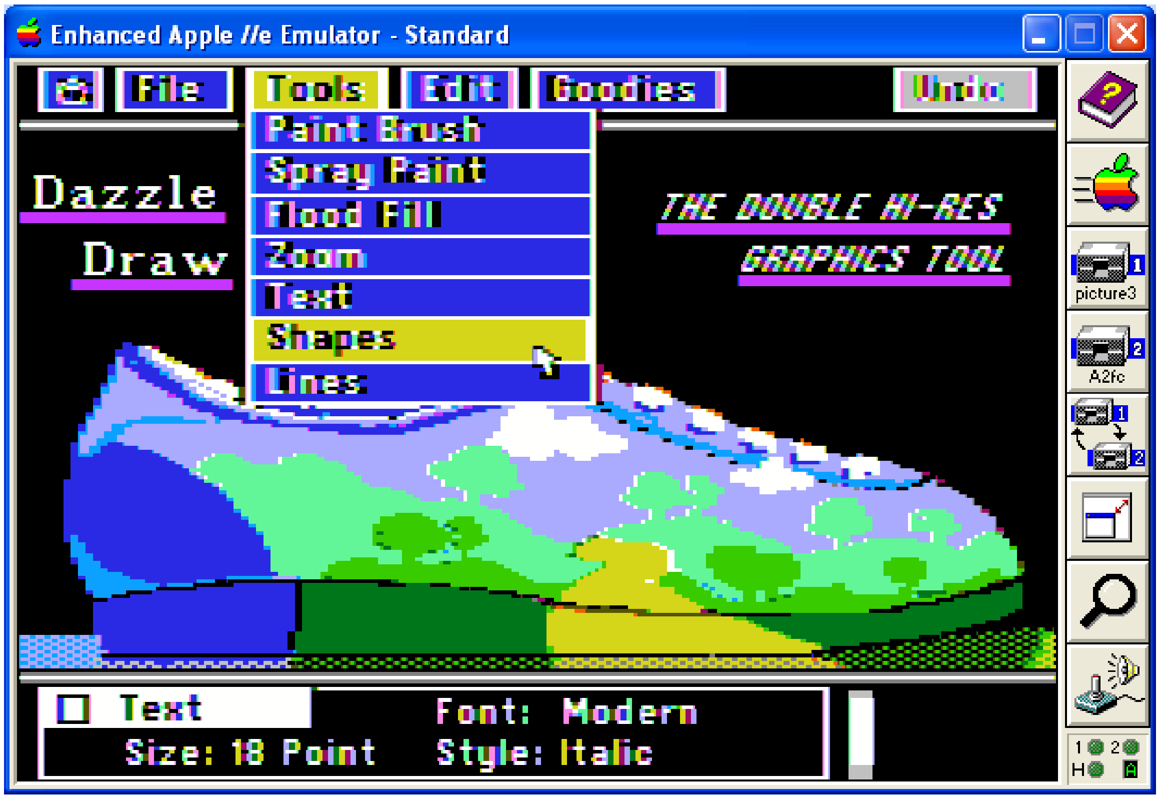
http://apple2history.org/

Unlike Unix/Linux and MS-DOS/Windows, Apple Computer did not start with the concept that there was only one type of file and 2 modes; text and binary. From before the time of DOS 3.3 on the Apple II, the notion of File Types and Auxiliary Types existed, and the notion of a transparent association of a Graphical Desk Top, and Data and Resource Forks which came quickly from Apple Computer, was far away from the minds of many Apple II users, the majority who were in some ways more hardware-savvy than the average CP/M and MS-DOS user. Unix was still on "big-iron" and barely accessible to home computer users, except through modem gateways and the university and other restricted access systems. This is a generalization of course since techies moved freely through any and all computers throughout all of computing history. But in general Apple II users were early accepters, as were early Mac users, and invested much time and patience into writing programs and other normal "Homebrew" activities.

After so many years of loyalty and devotion invested in their old computer, the Apple II user in particular saw little hurry to abandon the past, while Apple Computer tried to drag them forward into a future vision that was far ahead of its time. Many moved ahead but some more slowly, and some hardly at all:


KEGS32 - Apple //gs Emulator
Emulator Help
1 MHz
2.5 MHz
8 MHz
∞ MHz
KEGS 0.91 | Vol:5 | sim MHz:314.40 | Eff MHz: 8.17 | s6d1: 0 s6d2: 0 s5d1: 0/0 s5d2:15/0


Enhanced Apple //e Emulator - Standard
BRODERBUND SOFTWARE
DAZZLE DRAW
BY DAVID SNIDER
picture3
A2fc


Enhanced Apple //e Emulator - Standard
File
Tools
Edit
Goodies
Undo
Paint Brush
Spray Paint
Flood Fill
Zoom
Text
Shapes
Lines
Dazzle
Draw
THE DOUBLE HI-RES
GRAPHICS TOOL
Text
Font: Modern
Size: 18 Point
Style: Italic
picture3
A2fc

The Device Independent Bit Map was in wide use in the world at large, but on the old Apple II, bit-mapped graphics were still quite device dependent, so there was in general no need to change that. The early accepters left the old stuff behind. While Apple Computer attempted to provide a migration path forward to their Macintosh, many of their loyal Apple II users were still working and playing in DOS 3.3 and ProDOS 8 for quite a long time and even today. The phenomena of networks like CompuServe gave these same users more access to each other, and also to CompuServe's GIF file format.

One application of that day (1988), that is still available today, for converting GIF to DHGR files for use in ProDOS (files that could also be easily transferred to a DOS 3.3 disk), is a widely distributed freeware utility called "][gif" by Jason Harper:

http://nixietube.info/

"][gif" is the first (and only) 8 bit GIF converter for the Apple II. It saves DHGR files in A2FC format, but not ProDOS FileType $08. Apple II users in general have never (until recent years) needed to export less capable device-dependent DHGR files back to other more capable video platforms, and with converters like "][gif" can make the most of their old Apple II video modes. Today outside the Apple II not much about is remembered, except in limited accounts like the following, written from a modern perspective:

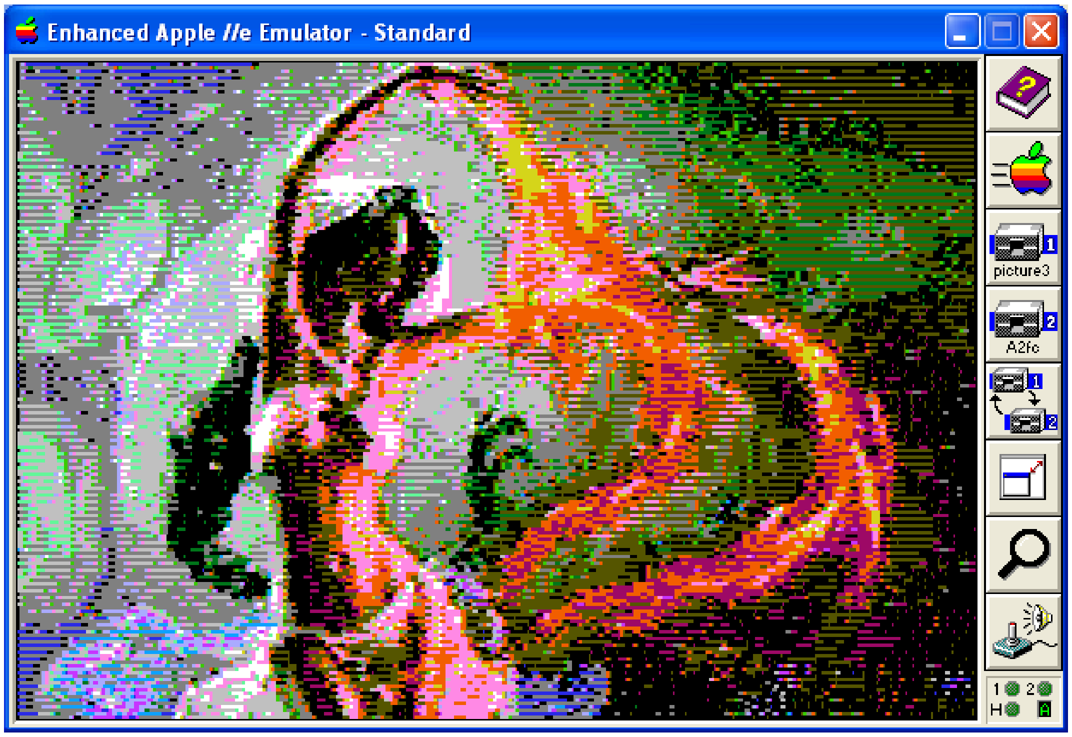
http://fileformats.archiveteam.org/wiki/Apple_II_graphics_formats

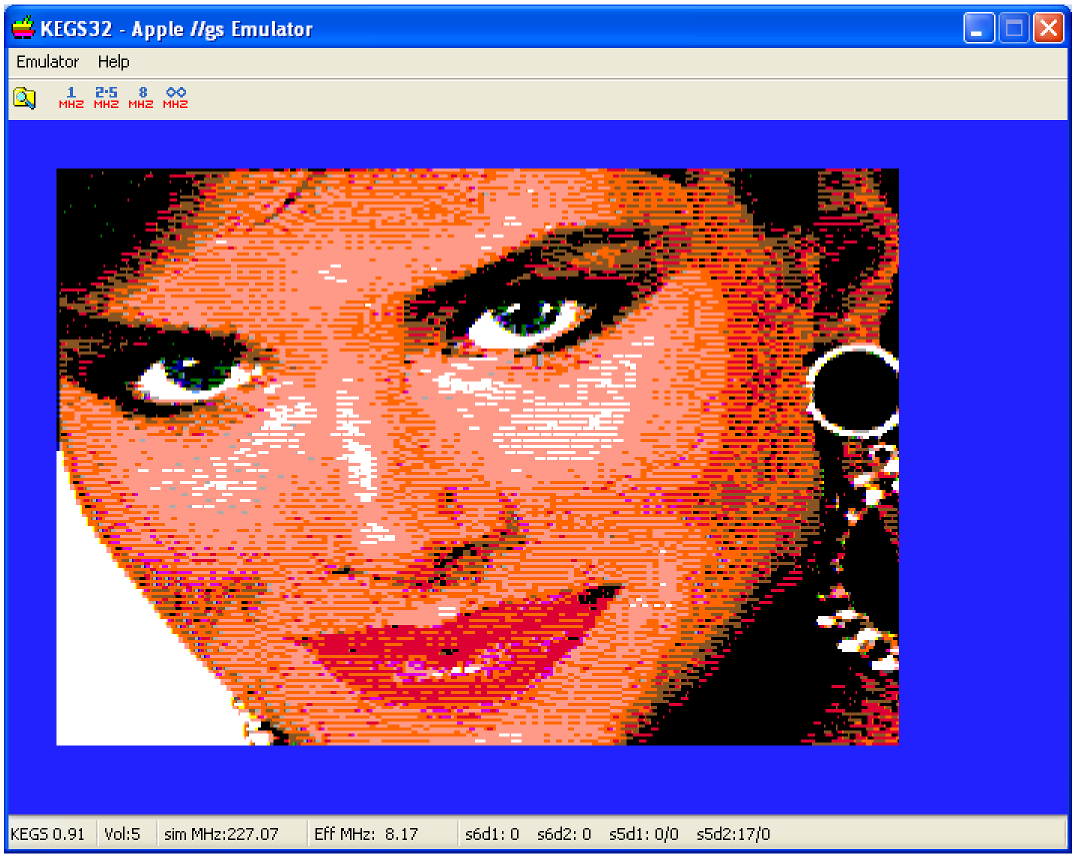
Many of the files in this document were converted using "][gif" after scaling from jpeg or BMP in Windows Paint, and then using MicroSoft Office Picture Manager to convert to GIF, and transferred to an Apple II disk image in CiderPress, followed by using "IIgif" to convert to A2FC. Scaling from 320 x 200, 44% horizontal, and 96% vertical, or 640 x 400, 22 % x 48% results in 140 x 192 (after clipping) and allows modern scaling and adjustment to be used before encoding and transforming to GIF. To convert from any other size make sure your aspect ration is more or less proportional to the Apple II screen. A circle should ideally still be circular at 280 x 192 on your screen when done. For all scaling prior to "IIgif" conversion the vertical percentage is 192/Y. For DHGR color, the calculation for horizontal percentage is 140/X and for HGR color, the calculation for horizontal percentage is 280/X. For Greyscale or Monochrome, just double the horizontal dividend. You can also use "IIgif" to scale, but pre-scaling on your modern computer results in smaller GIF files for "IIgif", and allows you to use other modern image tools on your desktop for better control of results.

So even today (especially) there is little need for the Apple II user to use ProDOS FileType $08 for DHGR, or even HGR files. "][gif" as always, converts GIF files to both "traditional" binary formats only.

"][gif" is available from **Asimov:**

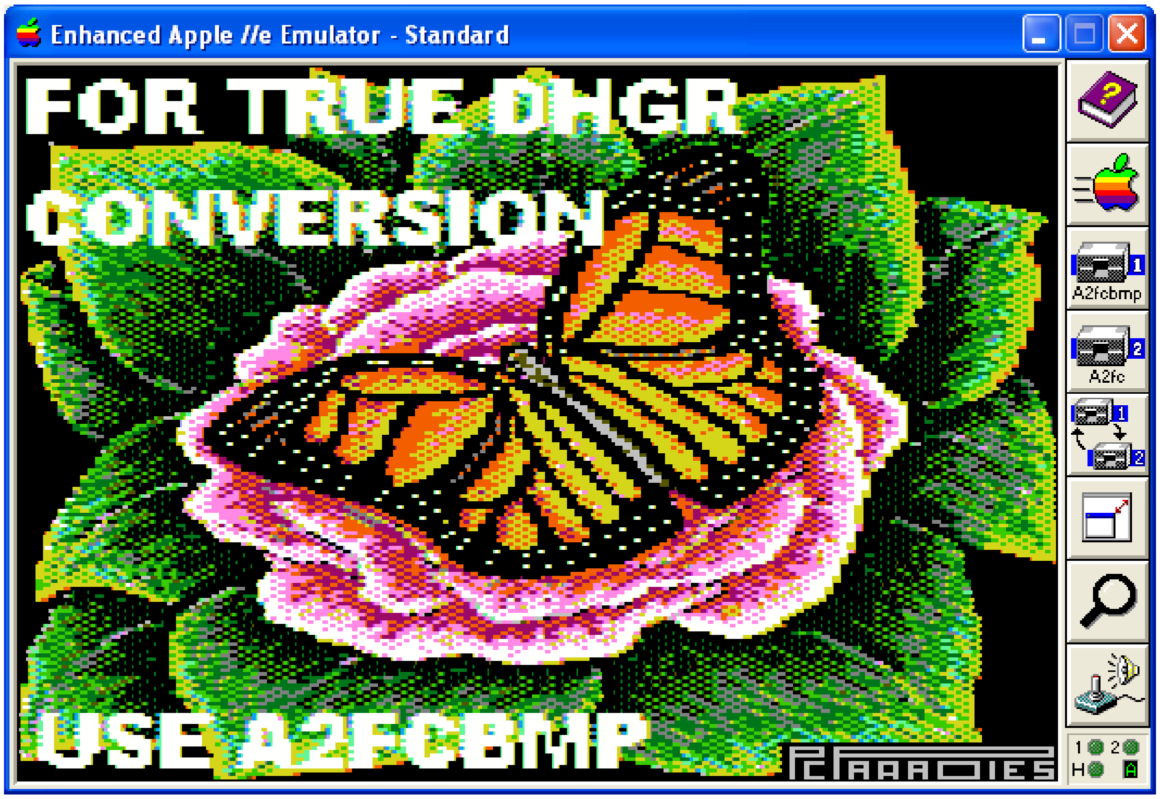
"][gif" Documentation
"][gif Program"





KEGS 0.91 | Vol:5 | sim MHz:227.07 | Eff MHz: 8.17 | s6d1: 0 s6d2: 0 s5d1: 0/0 s5d2:17/0

Supported DHGR Formats



Format File Size	Files	Description	Extension
AUX,BIN 8192 or 8184 bytes	2	Uncompressed DHGR Screen Memory Image BSaved Images DHGR Auxiliary and Main Memory	.AUX,.BIN
A2FC 16384 bytes	1	Uncompressed DHGR Screen Memory Image Concatenated BSaved Images - AUX,BIN http://whatis.techtarget.com/fileformat/A2FC-Image-Apple-IIe-Apple-IIc-Double-Hi-Res-16-writes-16-colors	.2FC

Back in 2009, as part of the larger collection of programming for the Apple II that the "dhishow" demo descends from, I wrote an MS-DOS utility called BMPA2FC.EXE. BMPA2FC interactively remaps the colors from a PC bitmapped graphic image then outputs 3 raw Apple II Double Hi-Res (DHGR) graphic image files in the 2 common formats noted in the table above.

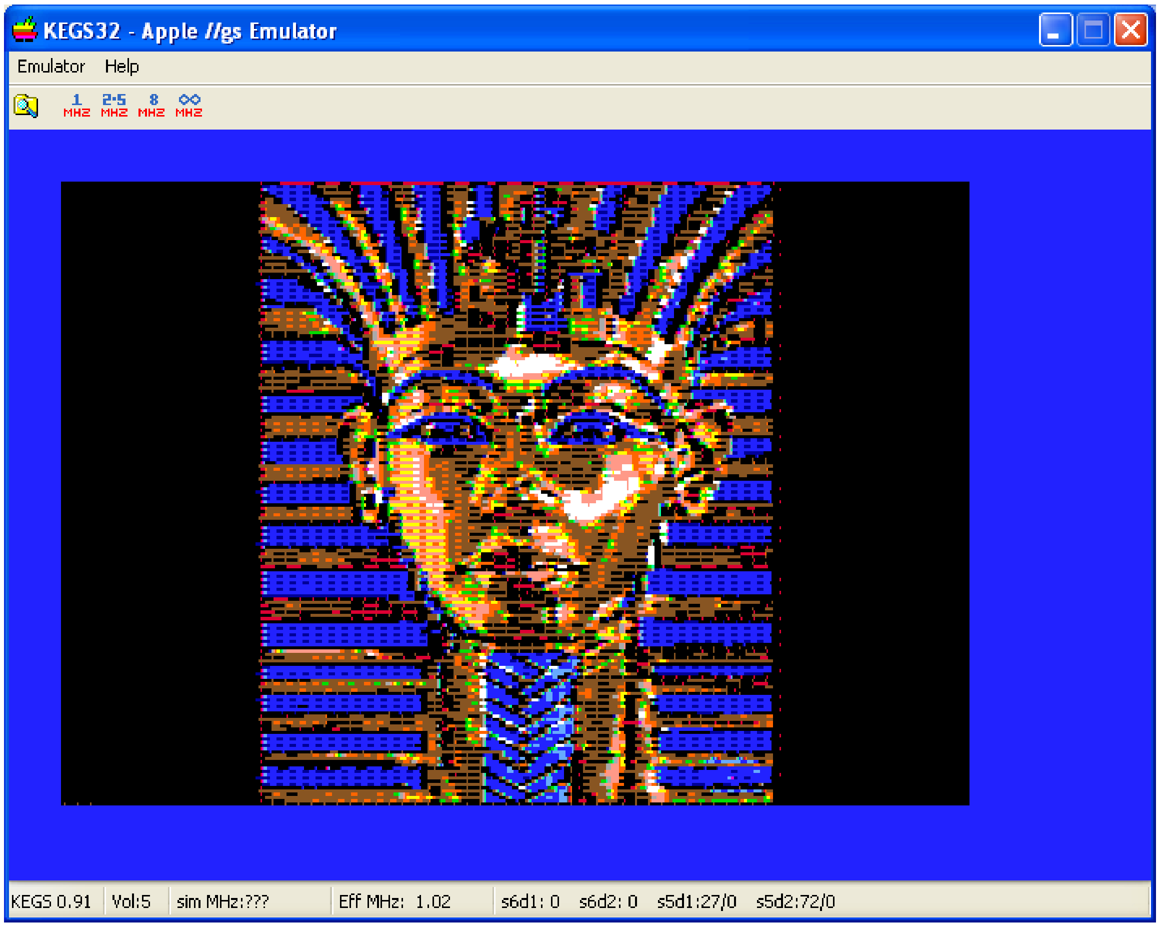
Because these raw files are a memory image of the DHGR Screen on the Apple II, so they can be displayed simply, by loading directly to DHGR Screen Memory without any decoding or reformatting.

AUX, BIN File Pairs are targeted towards AppleSoft BASIC programs where they can be BLOADed using the DOS 3.3 or ProDOS BLOAD command. Beagle Graphics also works with these file pairs, and names auxiliary memory files ".AUX" files. Sometimes these were BSAVEd to 8184 bytes. So this creates two sizes of AUX, BIN files; 8184 bytes which is the extent of the visible part of the screen, and 8192 bytes which is the extent of the screen's memory segment. A loader program must handle both sizes.

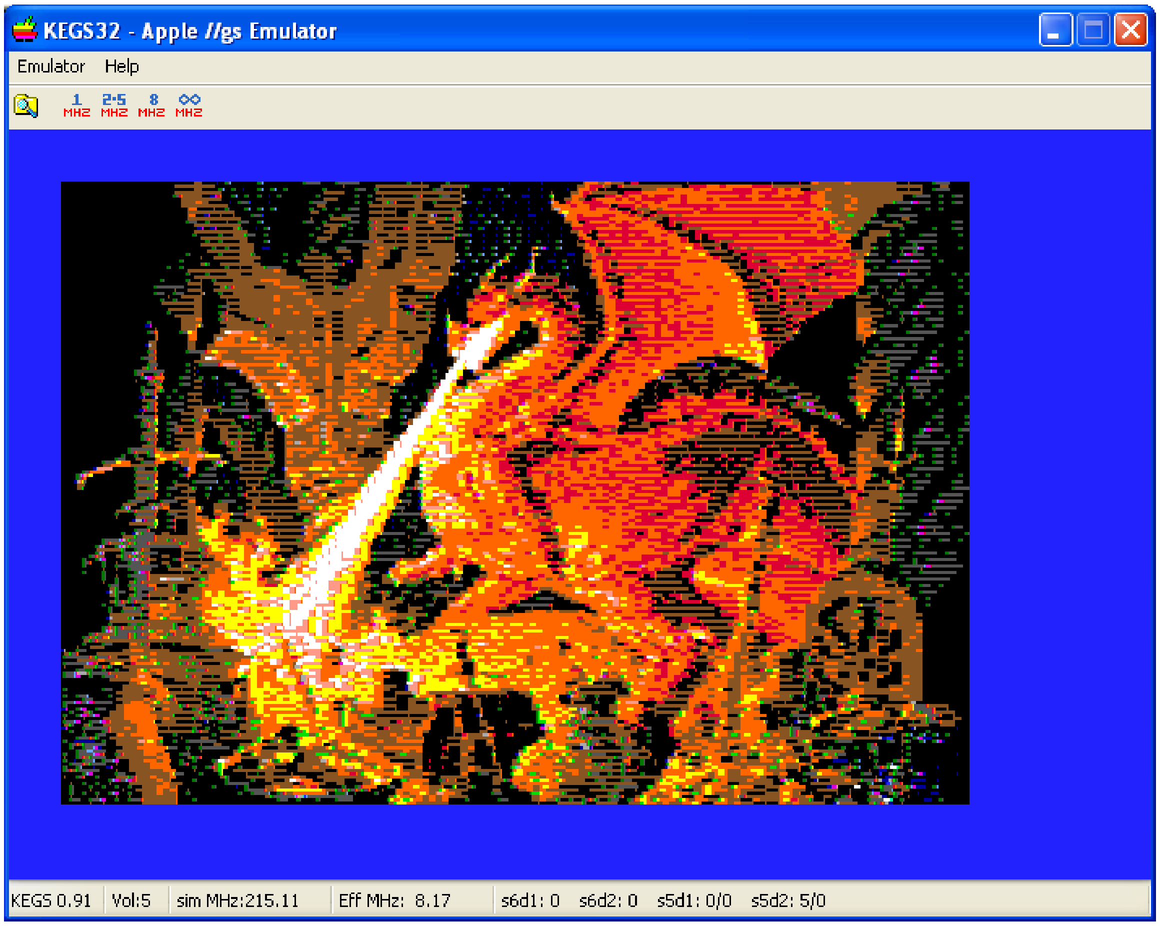
A2FC files are targeted towards later versions of ProDOS or applications written in higher level languages like David Snider's Dazzle Draw Paint Program (Broderbund Software, 1984) where they may have originated:

http://jmsteinerfilms.com/2013/01/04/qa-with-gaming-pioneer-david-snider/

But they were not called A2FC files in Dazzle Draw nor later (1988) in Jason Harper's "][gif" converter. Jason Harper's later Super Convert read these without needing a file extension, and then re-saved these as SHR (Super-Hi Res) files, yet another reason that ProDOS FileType $08 was not widely used for DHGR. And why anyone with a IIgs and of a serious nature, and with a ready supply of more capable graphics to convert would have wanted a DHGR file "back in the day" is also an interesting thought.



DHGR File Naming Conventions


KEGS32 - Apple //gs Emulator
Emulator Help
1 MHz 2.5 MHz 8 MHz ∞ MHz
KEGS 0.91 Vol:5 sim MHz:215.11 Eff MHz: 8.17 s6d1: 0 s6d2: 0 s5d1: 0/0 s5d2: 5/0

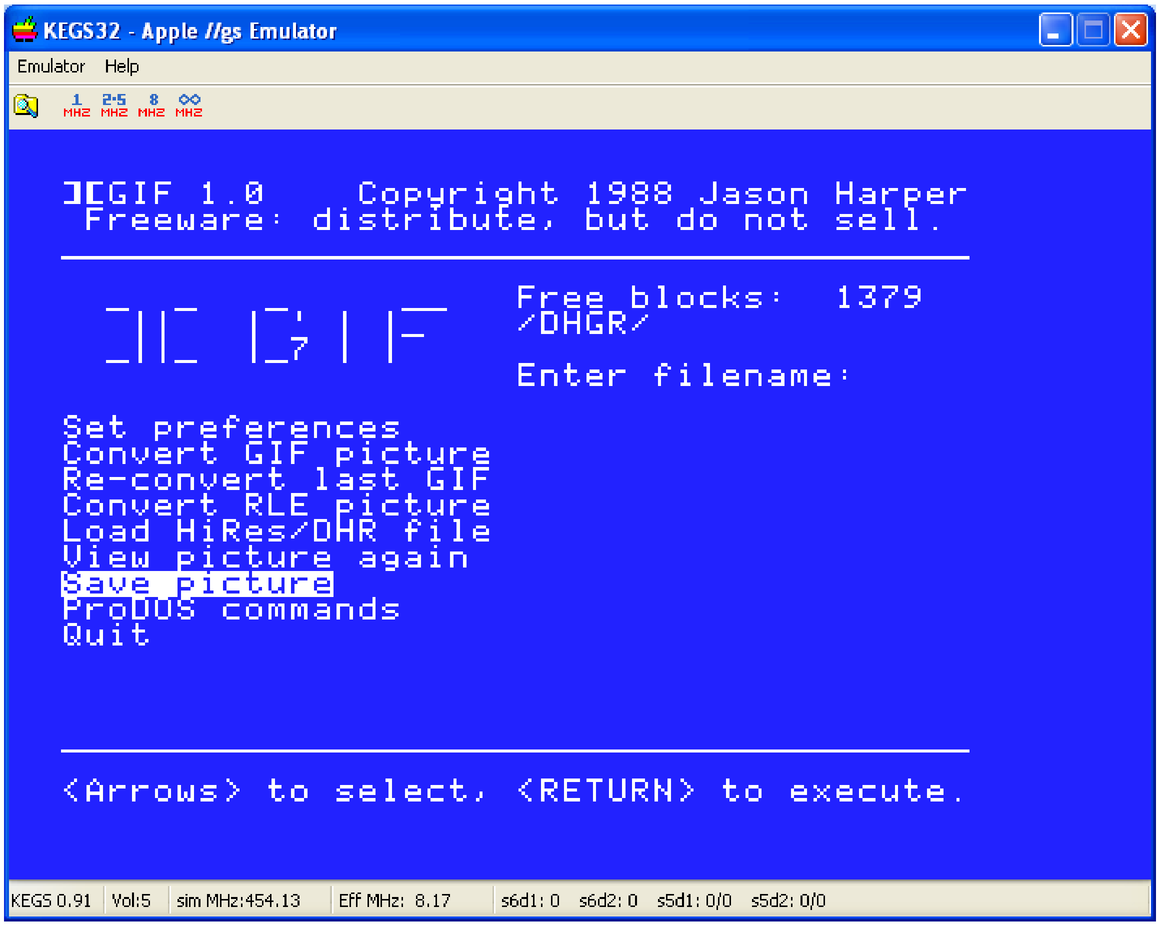
Because BMPA2FC is an MS-DOS program it uses the 8.3 file naming convention, hence the output file extensions .AUX, .BIN, and .2FC.

I am a big fan of automatic naming in my graphics converters and other programs. I generally use the “basename” of an input file, and automatically provide an extension based on the output file format. The idea is to keep these files grouped together in a sorted directory and for other reasons some of a lazy programming nature.

As noted, Beagle Graphics used the .AUX file extension for the BSAVED Auxiliary Memory file in the AUX, BIN file pair. I have always used the .BIN extension for an HGR (Hi-Res) BSaved File. In the interests of consistency, I adopted both naming conventions respectively for output of AUX, BIN file pairs in BMPA2FC.

While doing my DHGR work for Aztec C65, I came across a Stuffit archive, “II_Pictures.sit”, which contained several concatenated DHGR AUX, BIN files, dated 1995, with the extension .A2FC. Again in the interests of consistency for BMP2A2FC, I adopted this naming convention and shortened-it to .2FC to conform with MS-DOS.

But neither of these naming conventions means anything "official" when it comes to Apple II DHGR files. The Stuffit archive noted above ("II_Pictures.sit") also contains several concatenated DHGR AUX, BIN files with no extensions. Naming of AUX, BIN file pairs in the archive was also inconsistent. And as noted above, Stephen Harper's "][gif" file converter also allows "free form" naming but since GIF also follows 8.3 naming there is enough room in a ProDOS file name for a .BIN or .2FC extension if someone thought like an MS-DOS user instead of quite the opposite. The ProDOS filing system offered FileTypes and Auxiliary Types and former DOS 3.3 users felt the restrictions of moving from 30 character to 15 ProDOS's 15 character file names so weren't likely to much bother with CP/M's 8.3 naming convention used by MS-DOS.

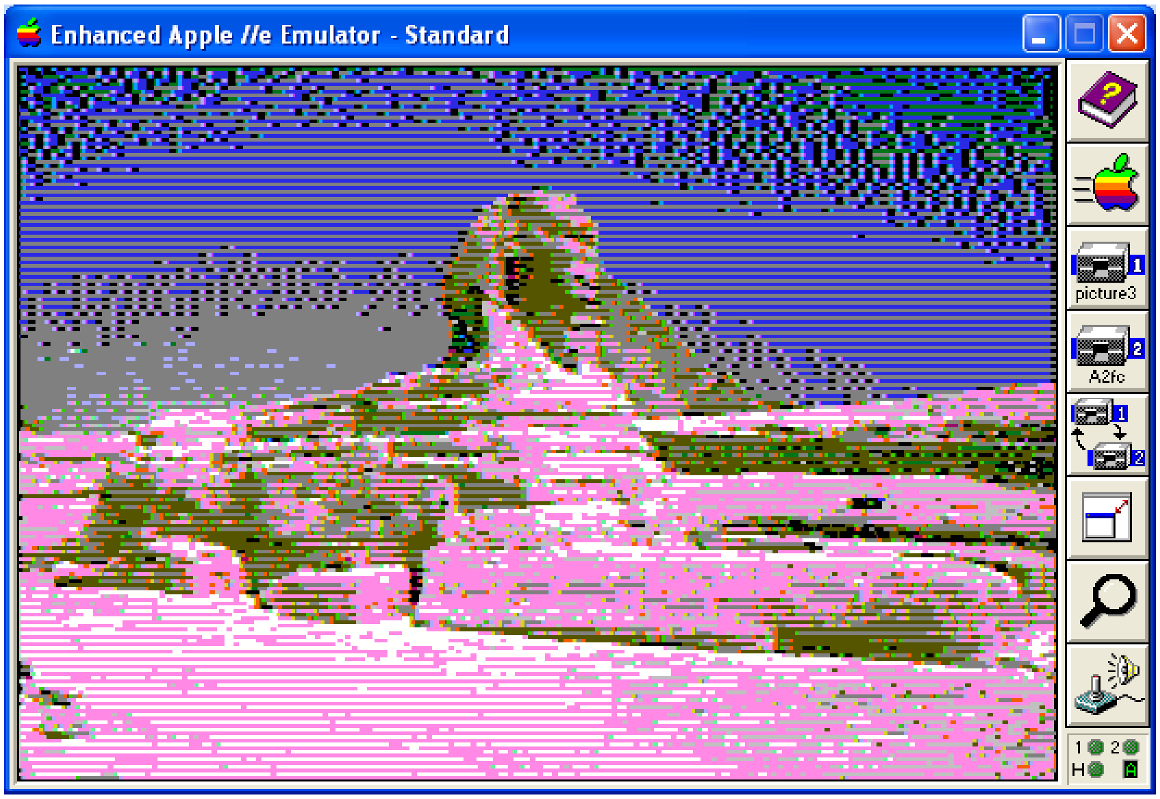


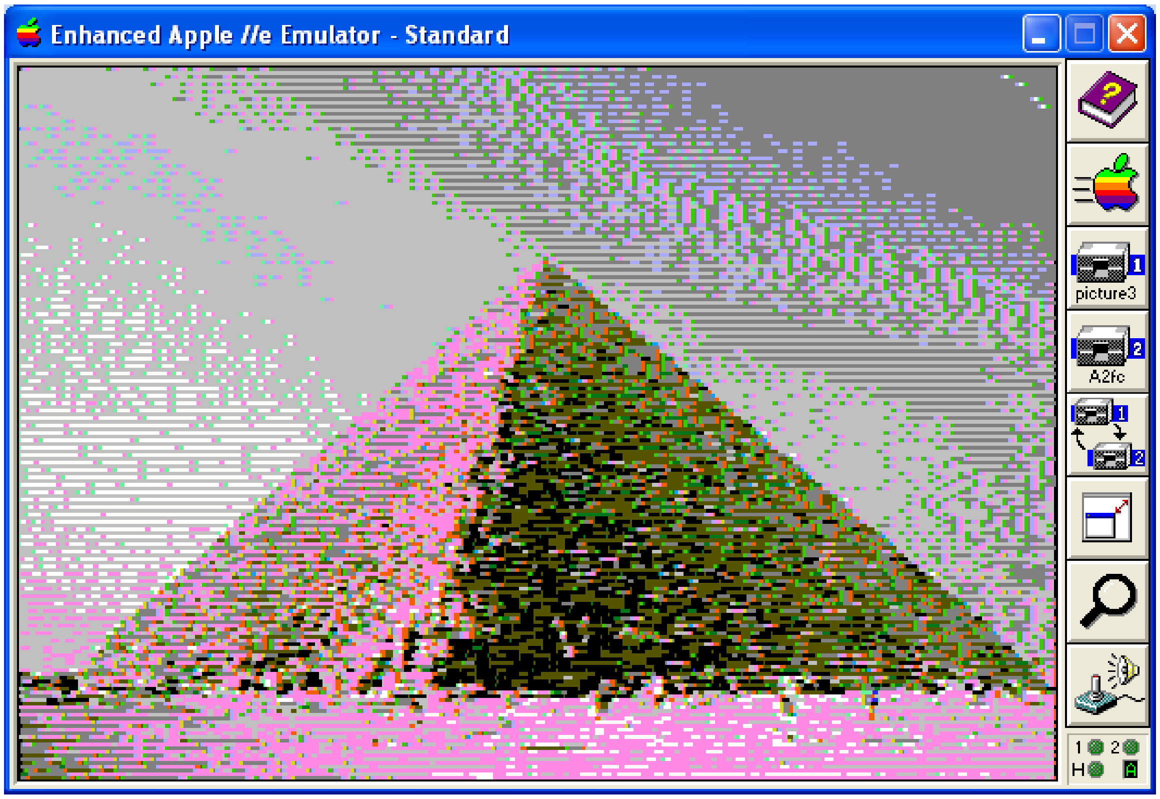
```
][GIF 1.0    Copyright 1988 Jason Harper
 Freeware: distribute, but do not sell.

                    Free blocks:   1379
                    /DHGR/

                    Enter filename:

Set preferences
Convert GIF picture
Re-convert last GIF
Convert RLE picture
Load HiRes/DHR file
View picture again
Save picture
ProDOS commands
Quit

<Arrows> to select, <RETURN> to execute.
```

KEGS 0.91 | Vol:5 | sim MHz:454.13 | Eff MHz: 8.17 | s6d1: 0 s6d2: 0 s5d1: 0/0 s5d2: 0/0

Also as noted above, "][gif" is available from **Asimov:**

"][gif" Documentation
"][gif Program"


Enhanced Apple //e Emulator - Standard
picture3
A2fc
1 2
H


Enhanced Apple //e Emulator - Standard
picture3
A2fc
1 2
H

Recent History

Writing BMPA2FC in 2009 was no problem, and writing Aztec C65 DHGR C programs in DOS 3.3 was no problem, so at first I thought I would have no problem displaying DHGR images in an Aztec C65 ProDOS SYS program. I was sadly mistaken, and after many failed attempts I became so frustrated with the Apple II and my Aztec C65 C compiler that I gave-up retro-programming altogether for over 2 years.

In March 2012 I was contacted by an Apple II retro-enthusiast by the name of Andrew Hogan who had "discovered" BMPA2FC on one of my websites while working with his son learning about DHGR. He suggested some additional features and I began to work with DHGR again, first expanding BMPA2FC, and then producing a companion utility called A2FCBMP and another called DMONO, followed quickly by solving whatever remaining problems I had with DHGR in ProDOS Aztec C65 programming

Taking a break had evidently worked. BMPA2FC and these other utilities became part of my Apple II Aztec C65 AppleX distribution's 2013 major release along with many demos including DHGR demos for the Apple IIe written in Aztec C65.

More about these utilities can be found at the following link:

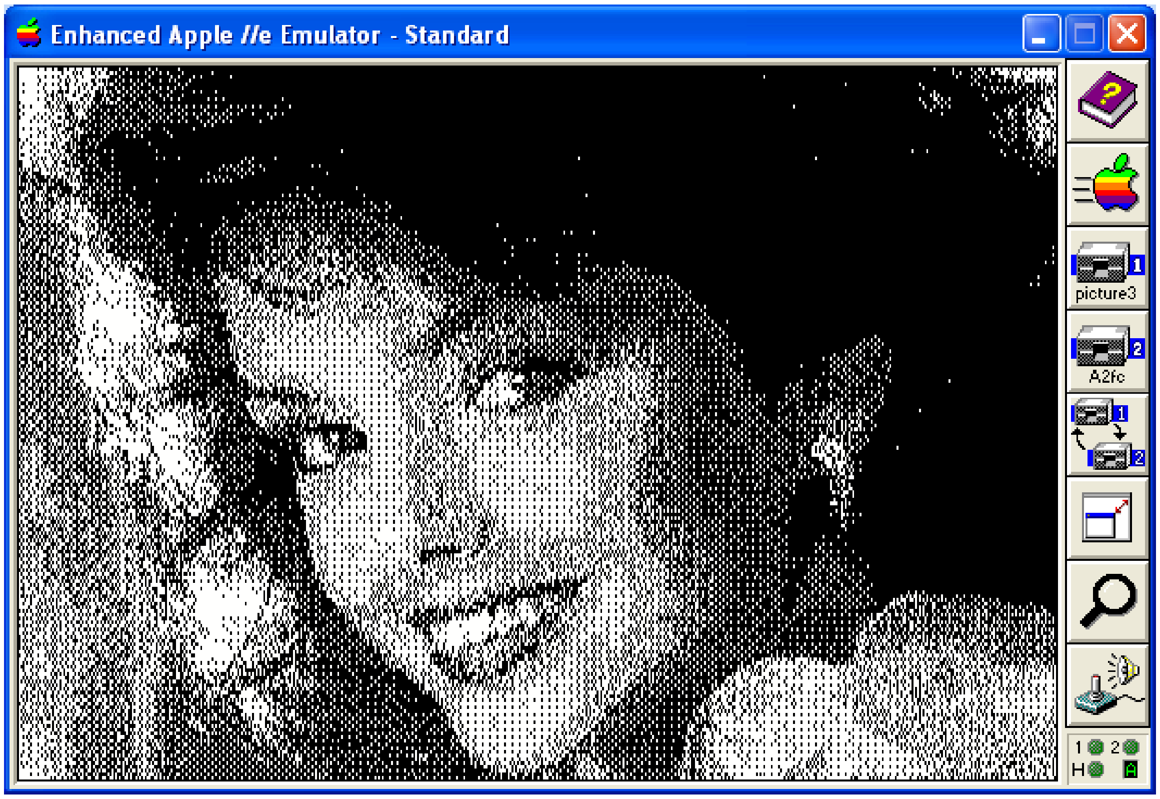
http://www.appleoldies.ca/graphics/index.htm

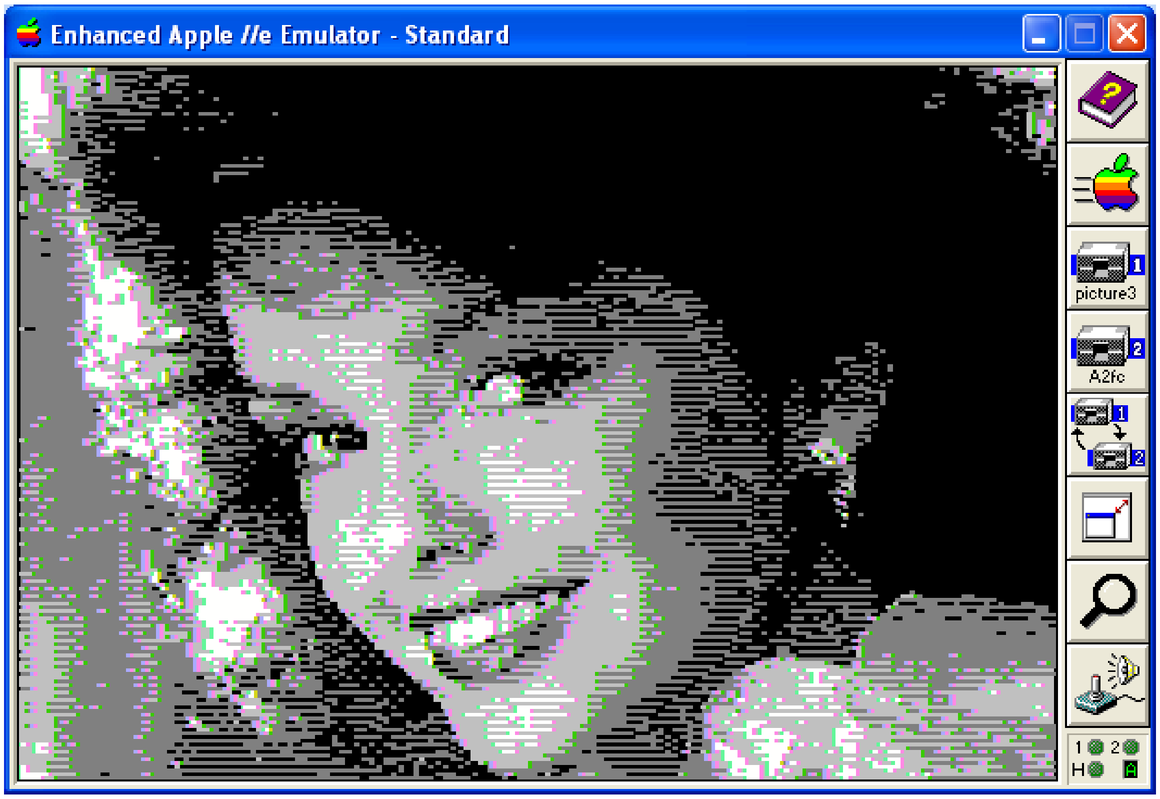
The Aztec C65 cross-compilers and most of the utilities I have written for them run in MS-DOS. Over the past year or so as I became an expert on the Apple II and legend in my own mind, I realized more and more the failings of Aztec C65 programs and Aztec C65 itself when compared to the cc65 cross-compiler. Becoming even more depressed than in 2009, I also realized more and more that MS-DOS is as dead as the Apple II, and even emulators like DOSBox don't always work with the oldest Aztec C65 compilers, like the very old Aztec C65 Commodore 64 compiler that I put together and distribute from the Aztec C Website:

http://www.aztecmuseum.ca

In 2009 I also decided to port all of my Aztec C65 work to the cc65 cross-compiler, but I wasn't quite ready. But by May 2014 I had decided to do my utilities exclusively in the MinGW gcc compiler where possible, and abandon MS-DOS where possible. It was time to port everything else related to programming the Commodore 64 and Apple II to cc65.

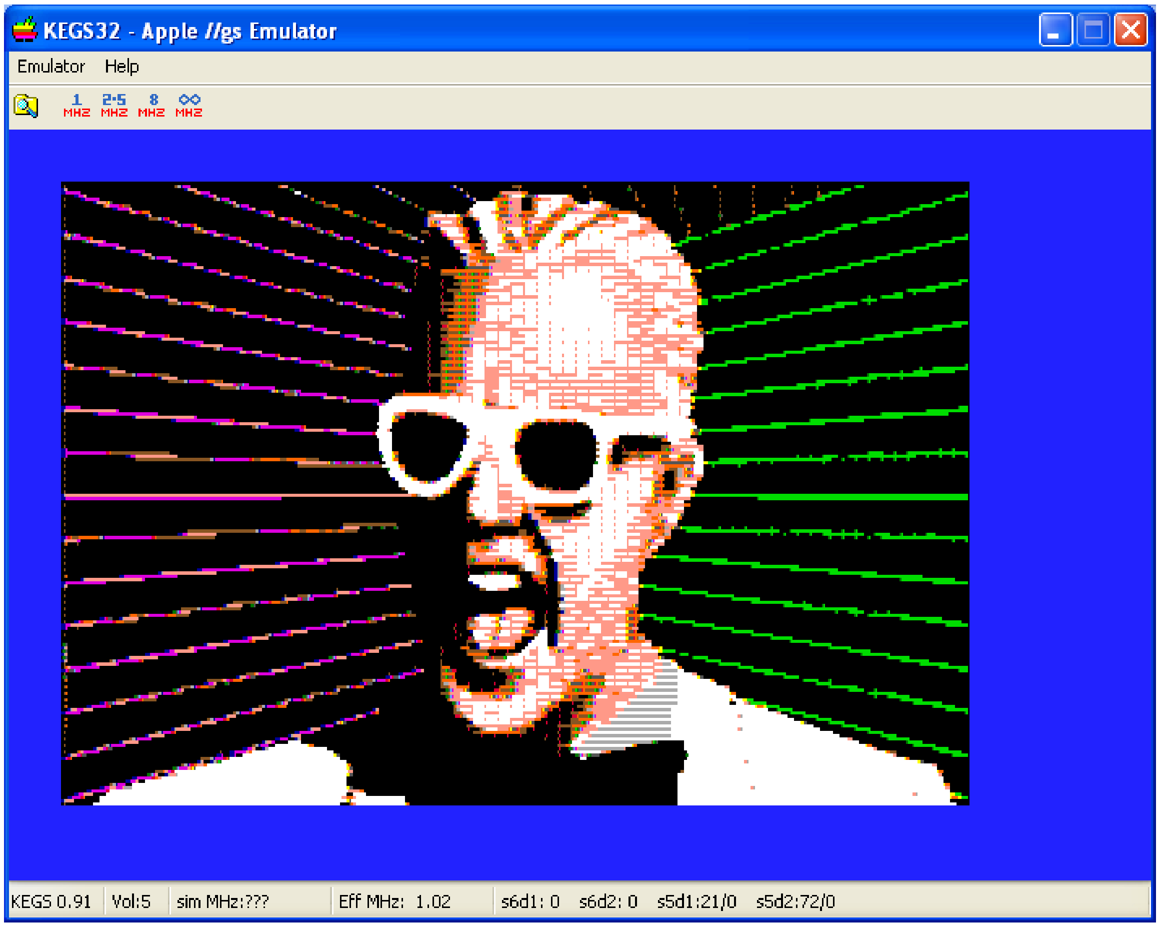
MS-DOS support is quickly vanishing from the planet. Cc65 writes generally faster and smaller code than Aztec C65, is actively developed and seems to run everywhere, so porting of dhishow from last year's AppleX major release to cc65 comes none too soon.


Enhanced Apple //e Emulator - Standard
picture3
A2fc


Enhanced Apple //e Emulator - Standard
picture3
A2fc

Introduction - DHISHOW in cc65

HeadRoom

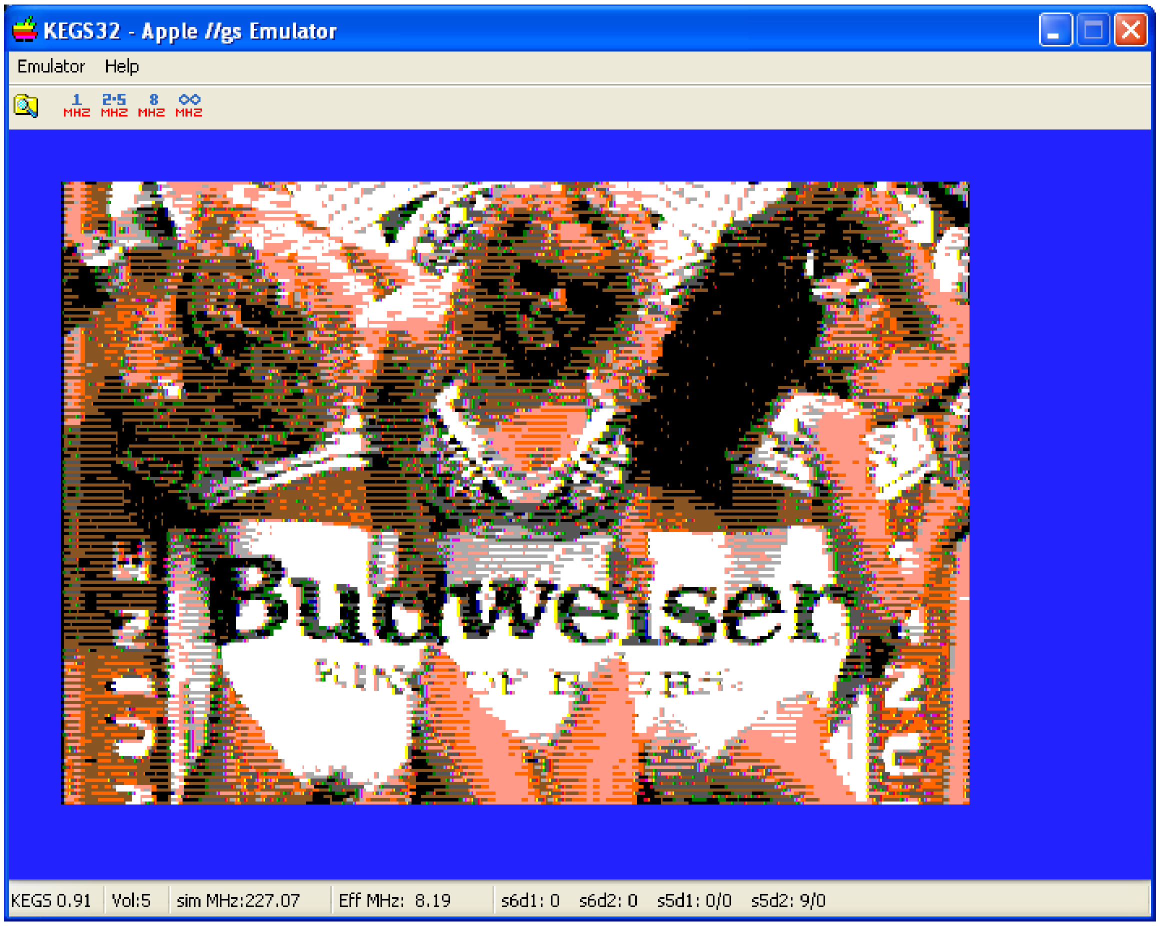

KEGS32 - Apple //gs Emulator
Emulator Help
1 MHz 2.5 MHz 8 MHz ∞ MHz
KEGS 0.91 Vol:5 sim MHz:??? Eff MHz: 1.02 s6d1: 0 s6d2: 0 s5d1:21/0 s5d2:72/0

The dhishow.c source code produces two executables; dhishow and dhishow2. These two executables are practically the same and share a common source file, a common MAKE file but not a common linker configuration file. Dhishow runs at $4000 and uses Screen Page One DHGR at $2000 directly beneath the program area. Dhishow2 runs at $0803 and uses Screen Page Two DHGR at $4000. Dhishow2 will run out of memory before dhishow because it has very little headroom, only 14333 bytes of program memory, compared to the 30464 bytes of program memory available to the dhishow binary.

Either variation of dhishow has sufficient memory for this demo program because the code is very simple. The dhishow cc65 binaries are only about 9K and almost identical in size.

But there really isn't much point in dhishow2 except as a simple slideshow demo and to explore cc65's linker configuration scripts a little. Dhishow2 is the opposite of prescriptive and even the dhishow binary is not the best way ever; just a demo.

DHSHOW Program Execution


KEGS32 - Apple //gs Emulator
Emulator Help
1 MHz 2.5 MHz 8 MHz ∞ MHz
Budweiser
KEGS 0.91 Vol:5 sim MHz:227.07 Eff MHz: 8.19 s6d1: 0 s6d2: 0 s5d1: 0/0 s5d2: 9/0

When DHISHOW starts it checks the current directory for a slide-show script called "PICLIST"; an Apple II format sequential text file with a DHGR file name on each line. If PICLIST is not found, it is automatically created from all the available DHGR files in the current directory. A PICLIST can be created or revised and edited in any text editor that works with Apple II text. (DHISHOW will also read a PICLIST in "unix" or Windows format if you prefer.)

DHISHOW then goes on to display the DHGR slide-show. The DHGR "slides" display continually. When the end of PICLIST is reached it starts over at the beginning. The ESC key can be pressed at any time to exit DHISHOW and any other key will just advance to the next slide. By default, if no key is pressed DHISHOW displays a slide for 5 seconds using a timing loop based on a 1 MHZ Apple II, just like a Beagle Bros. slide-show.

DHISHOW Delay

On the Apple II you can have a clock add-on like the no-slot clock (NSC) but unlike the Commodore 64's TOD (Time of Day) Clock which is built-in, we have no such beast that is consistent across all Apple IIs; clocks like the NSC are yet another afterthought with drivers in memory, so dhishow uses a simple timing loop based on a 1 MHZ Apple II.

By default DHISHOW waits for 5 seconds between loading of slides. A text file called "DELAY.TXT" in Apple, "unix", or Windows format can be placed in the current working directory, with a single line that provides the delay in seconds on a 1 MHZ machine; placing the value of "10" in this file changes the delay from the default of 5 seconds to 10 seconds.

For a faster Apple II, scale that value by multiplying by MGZ. On a 2.5 MGZ Apple II enter 25 for 10 seconds, and on an 8 MHZ machine enter 80 for 10 seconds. Scaling may need some fine tuning if you want pinpoint accuracy.

If delay is set to 0, then the timing loop is bypassed and the slide-show must be advanced by a key press.

Program Header

```
/* ------------------------------------------------------------------
System       : cc65 cross-development environment
Platform     : Apple IIe 128K PRODOS 8
Program      : dhishow(C) Copyright Bill Buckels 2013,2014.
               All rights reserved.

Description  : Apple //e ProDOS 8 DHGR Slideshow Program

               DHGR formats supported: A2FC and AUX, BIN file pairs

               This program builds two executables based on conditional
               compilation. The code for either is almost exactly the
               same, but they use two different memory configurations
               based on whether they use DHGR Screen One or Screen Two.

               The dhishow variation that uses Screen One runs at $4000
               directly above Screen One at $2000 and has a good deal
               more memory than the dhishow variation that uses Screen
               Two which runs at $0803 under Screen Two (at $4000).

               Either can be launched using Oliver Schmidt's
               loader.system or from BASIC.SYSTEM with the BRUN
               command.

Date Written : February 2013
Revision     : 1.0 First Release (Aztec C65)
               Originally released as dpshow
Date Revised : May 2014
               2.0 Second Release (cc65) - stripped-down version.

Licence      : You may use this program for whatever you wish as long
               as you agree that Bill Buckels has no warranty or
               liability obligations whatsoever from said use.
------------------------------------------------------------------ */
```

```
#ifndef DHGR_SCREEN_ADDRESS
#define DHGR_SCREEN_ADDRESS 0x2000
#else
#if DHGR_SCREEN_ADDRESS != 0x4000
#define DHGR_SCREEN_ADDRESS 0x2000
#endif
#endif

#ifndef __APPLE2__
#define __APPLE2__
#endif

#include <stdio.h>
#include <errno.h>
#include "_file.h"
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <conio.h>

/* http://www.unix.com/man-page/posix/3/opendir/ */
/* files in current directory */
#include <dirent.h>

#include <apple2enh.h>
```

Note the above conditional compilation directive for DHGR_SCREEN_ADDRESS. This directive makes it possible to use the same dhishow.c program for both variations of memory configuration. You may occasionally hear reasons for not using conditional compilation in a C program. Like any language feature in C, if conditional compilation were not part of the language then compilers would not support it. Conditional directives are the very cornerstone of re-usable code, but careful thought must also be given to their placement to both minimize their use by organizing the code properly, and to minimize the amount of code affected, and they should where possible be defined both in the MAKE file, and also by defaults provided where needed. It is only by practice that this skill can be learned, so avoiding their use makes no sense at all unless a programmer is timid or sloppy or both.

Naming of constants for conditional compilation should be upper case and generally more descriptive than enums, ordinals, and the like which are also best upper-cased. Macros on the other hand should follow the same naming as functions where possible unless some cultural reason exists, like cc65's POKE in <peekpoke.h> which tries to remind you of BASIC.

Overly long names in constants in general is a relatively new phenomena because in older compilers which ran in lower memory environments, symbols were kept to a minimum size and are sometimes truncated by the compiler in self-defence, which also causes problems of truncated too-long handles and prefixes for Camel-Cased naming creating duplicate symbols. Style in general is a personal choice but best practices also exist.

Core Routines

What follows now are the core routines. Dhishow does not have its own header file since it is so simple. Functions, defines, and variables are placed ahead of where they are used.

Delay Routine

```
/* 1MHZ timing loop settings - increase for faster machines */
#define YTIME 60L
#define XTIME 32L
char wait(unsigned duration)
{
    char c = 0;
    long y, x;

    while (duration > 0) {
        for (y=0;y<YTIME;y++) {
            for (x=0;x<XTIME;x++) {
                if (kbhit()) {
                    c = cgetc();
                    /* clear keyboard buffer */
                    while (kbhit())cgetc();
                    return c;
                }
            }
        }
        duration--;
    }
    return c;
}
```

Slide-Show Script Builder

```
/* the following function creates a list of the double hi-res
   files in the current directory, in Apple sequential text format. */
FILE *makepiclist()
{
    FILE *fp = NULL;
    int err, cnt = 0, idx, jdx, found;
    DIR *dir;
    struct dirent *dp;

    extern unsigned char _filetype;
    extern unsigned _auxtype;

    /* read files in current directory */
    if ((dir = opendir (".")) == NULL) return NULL;

    /* remove piclist if it already exists - should not happen */
    if((fp=fopen("PICLIST","r"))!=NULL) {
        fclose(fp);
        remove("PICLIST");
    }
```

To set the ProDOS FileType and Auxiliary Type we need to go outside the C programming language's normal file operations and into the ProDOS MLI (Machine Language Interface). The cc65 MLI support is not exposed to the user, but the Create Block structure in the cc65 MLI layer exports data that can be used to set the FileType and Auxiliary Type before calling ProDOS to create a file:

```
/* set file to sequential text */
_filetype = 0x04;
_auxtype = 0x0000;

if((fp=fopen("PICLIST","w"))==NULL) {
    closedir(dir);
    return NULL;
}
```

As we read through the ProDOS directory file, cc65 provides us with extended information about files in a directory using an extended POSIX-like File Entry in the form of the dirent structure which contains ProDOS specific information as well as file length in bytes. We are primarily looking for Binary Files (ProDOS FileType $06) with a Load Address (Auxiliary Type) of $2000 and $4000, the base address of either DHGR screen. We are also looking for full screen DHGR FOT files (ProDOS FileType $08) in A2FC data format.

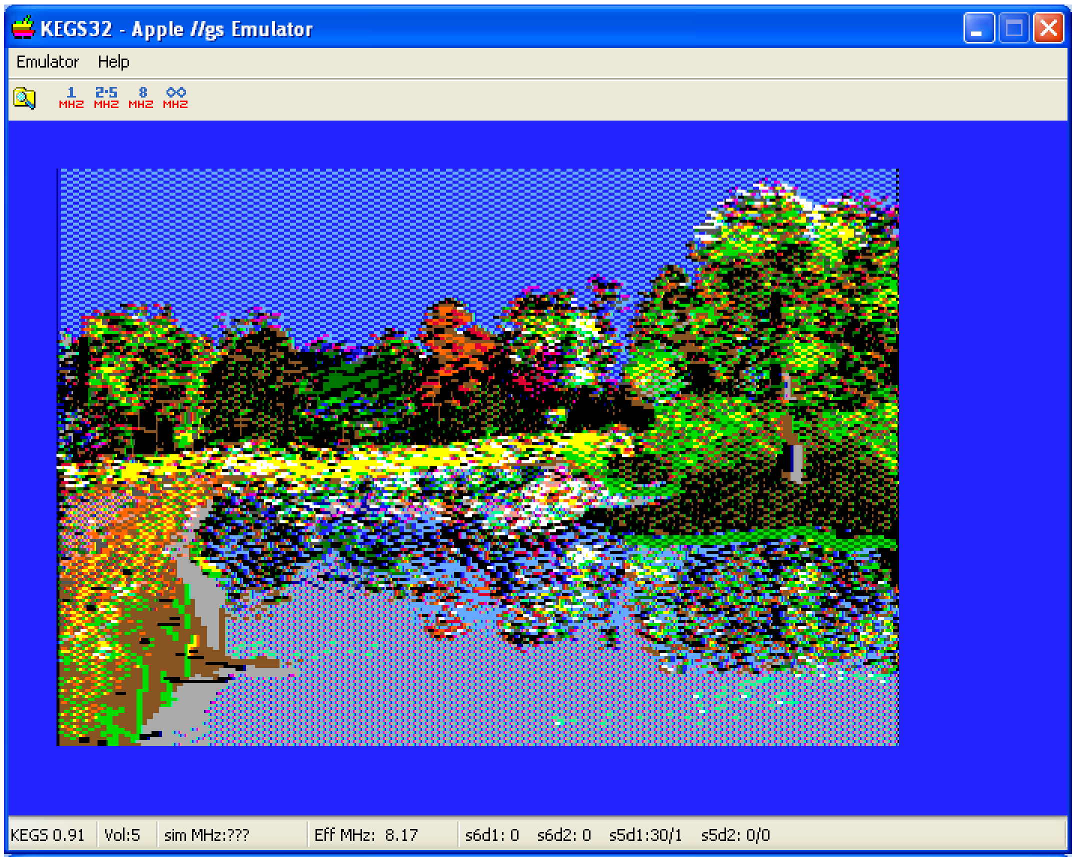
We are careful about not putting any duplicate file "basename" with the extension of .AUX into the PICLIST because all AUX files in AUX, BIN file pairs require the .AUX extension (in dhishow) whereas BIN files and A2FC files do not require an extension.. Other than that, we are not enforcing any naming conventions. HGR BIN files are indistinguishable from DHGR AUX and BIN files anyway, so if this is run with HGR files in the directory, they will go in here too, like swiss cheese.

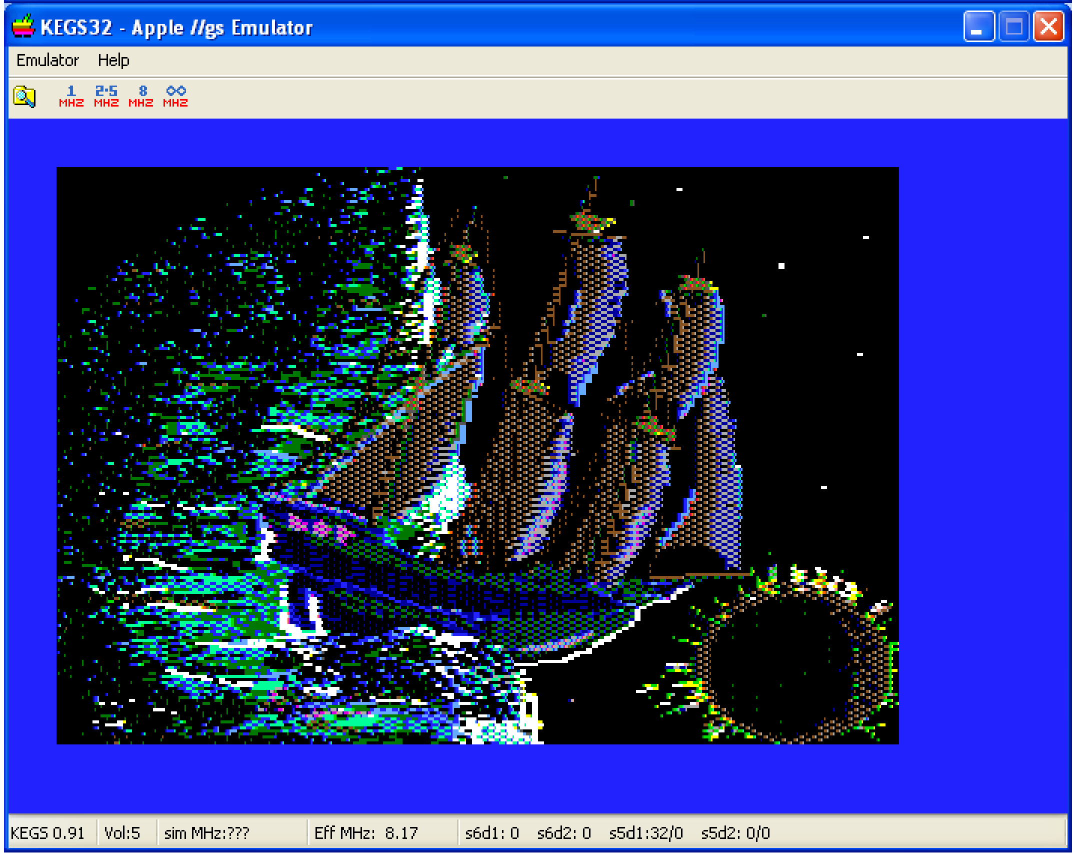
```
/* write the list to disk */
while ((dp = readdir (dir)) != NULL) {
    /* check for a2fc images and bin and aux image pairs */
    /* binary and FOT files only */
    if (dp->d_type == 0x08) {
        /* bypass all FOT files except for A2FC style files */
        if (dp->d_auxtype > 0x3fff) continue;
        if (dp->d_size != 16384) continue;
    }
    else {
        if (dp->d_type != 0x06) continue;
        /* load address 0x2000 or 0x4000 */
 if (dp->d_auxtype  != 0x2000 && dp->d_auxtype != 0x4000) continue;
        /* check file size */
        /* there is no way to tell the difference between a
        hi-res bsaved image and a double hi-res bin or aux image */
        found = 0;
        if (dp->d_size > 8183  && dp->d_size < 8193) found = 1;
        if (dp->d_size > 16375 && dp->d_size < 16385)found = 1;
        if (found == 0) continue;
    }
    /* get extension if any */
```

```
        jdx = 999;
        for (idx = 0; dp->d_name[idx] != 0; idx++) {
            if (dp->d_name[idx] == '.') {
                jdx = idx+1; break;
            }
        }
        if (jdx != 999) {
            found = 0;
            for (;;) {
            /* bypass AUX files - I only want BIN files in my list */
                if (toupper(dp->d_name[jdx+1]) != 'A') break;
                if (toupper(dp->d_name[jdx+2]) != 'U') break;
                if (toupper(dp->d_name[jdx+3]) != 'X') break;
                found = 1;
                break;
            }
            if (found == 1) continue;
        }
        /* if out of disk space quit writing to piclist */
        err = fprintf(fp,"%s%c",dp->d_name, (char)13);
        if (err < 0) break;
        cnt++;
    }
```

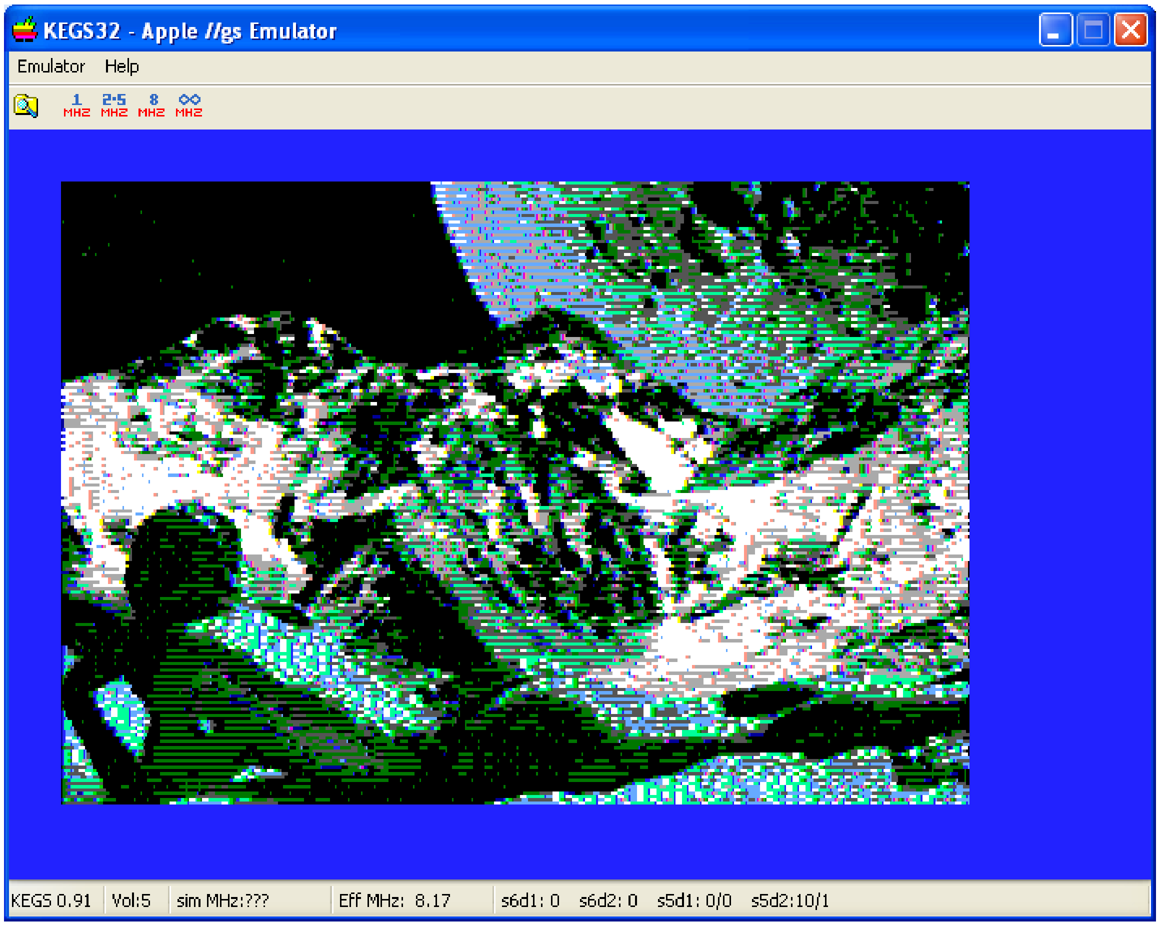
Note above that we are creating an Apple II native sequential text file with carriage returns and not a “unix” text file. The Apple II never used “unix” text natively.
Now that we are done reading the current directory and we have built the PICLIST we close the file and the directory. If no DHGR files were found, we return a NULL file pointer. Otherwise we return a file pointer to an open PICLIST to start the slide-show:

```
    fclose(fp);
    closedir(dir);

    if (cnt == 0) {
        /* if no files found or no filenames written
        remove empty piclist */
        remove("PICLIST");
        fp = NULL;
    }
    else {
        /* otherwise open and return the piclist */
        fp = fopen("PICLIST","r");
    }

    return fp;
}
```


KEGS32 - Apple //gs Emulator
Emulator Help
1 MHz 2.5 MHz 8 MHz ∞ MHz
KEGS 0.91 Vol:5 sim MHz:??? Eff MHz: 8.17 s6d1: 0 s6d2: 0 s5d1:30/1 s5d2: 0/0


KEGS32 - Apple //gs Emulator
Emulator Help
1 MHz 2.5 MHz 8 MHz ∞ MHz
KEGS 0.91 Vol:5 sim MHz:??? Eff MHz: 8.17 s6d1: 0 s6d2: 0 s5d1:32/0 s5d2: 0/0

Reading the PICLIST for Apple II Compatibility


KEGS32 - Apple //gs Emulator
Emulator Help
1 MHz 2.5 MHz 8 MHz ∞ MHz
KEGS 0.91 Vol:5 sim MHz:??? Eff MHz: 8.17 s6d1: 0 s6d2: 0 s5d1: 0/0 s5d2:10/1

I mentioned above that cc65 does not handle Apple II text files, so we replace the library call fgets() with a local fgets() slightly modified from the cc65 library version and the cc65 linker uses ours and not theirs for reading PICLIST when we display the slide-show.

```
/* fgets for Apple II text files - read native apple II text files
properly! The cc65 version doesn't! Which means that you need to use
unix text files unless you roll your own fgets like this one which does
both... */
char* __fastcall__ fgets (char* s, unsigned size, register FILE* f)
{
    register char* p = s;
    unsigned i;
    int c;

    if (size == 0) {
        /* Invalid size */
        return (char*) _seterrno (EINVAL);
    }

    /* Read input */
    i = 0;
    while (--size) {

        /* Get next character */
        if ((c = fgetc (f)) == EOF) {
            /* Error or EOF */
            if ((f->f_flags & _FERROR) != 0 || i == 0) {
                /* ERROR or EOF on first char */
                *p = '\0';
                return 0;
            } else {
                /* EOF with data already read */
                break;
            }
        }

        /* One char more */
        *p = c;
        ++p;
        ++i;

        /* Stop at end of Apple II or Unix Text File line */
        if ((char)c == (char)13 || (char)c == (char)10) {
            break;
        }
    }

    /* Terminate the string */
    *p = '\0';

    /* Done */
    return s;
}
```

Video Routines

Keep in mind that since we use the language card memory for other purposes in our linker config, we must use cc65's routine to safely set to 80 column mode.

```
/* 80 column mode must be set before calling */
void dhireson(void)
{
    /* page 1 or 2 double hires */
    asm("sta $c000"); /* TURN OFF 80 STORE        */
    asm("sta $c050"); /* GRAPHICS                 */
#if DHGR_SCREEN_ADDRESS == 0x4000
    asm("sta $c055"); /* PAGE TWO                 */
#else
    asm("sta $c054"); /* PAGE ONE                 */
#endif
    asm("sta $c052"); /* GRAPHICS ONLY, NOT MIXED */
    asm("sta $c057"); /* HI-RES                   */
    asm("sta $c05e"); /* TURN ON DOUBLE RES       */
}
```

The above is conditionally compiled to work for either DHGR Screen Address.

```
void dhiresclear(void)
{
    /* clear main screen memory */
    memset((char *)DHGR_SCREEN_ADDRESS,0,8192);
    /* clear auxiliary screen memory */
    maintoaux(DHGR_SCREEN_ADDRESS,DHGR_SCREEN_ADDRESS+8191,8192);
}

/* 80 column mode must be set to on after calling */
void dhiresoff(void)
{
    asm("sta  $c051"); /* TEXT - HIDE GRAPHICS */
    asm("sta  $c05f"); /* TURN OFF DOUBLE RES  */
    asm("sta  $c054"); /* PAGE ONE             */
    asm("sta  $c001"); /* TURN ON 80 STORE     */
}
```

Auxiliary Memory Routine

```
/* move a block of data from main to auxiliary memory */
void maintoaux(unsigned src0, unsigned src1, unsigned dest0)
{
    unsigned *src  = (unsigned *)0x3c;
    unsigned *dest = (unsigned *)0x42;

    src[0] = src0;
    src[1] = src1;
    dest[0] = dest0;

    asm("sec");
    asm("jsr $c311");
}
```

AUXMOVE is generally a handy routine for any Apple II program that stores and retrieves data in auxiliary memory, and must be called with 80Store off. The carry flag determines the direction of the move. This routine is called in two places in this demo; to clear the half of the DHGR Screen that is in Auxiliary Memory and to move the DHGR Screen Data from the file read buffer into the DHGR Screen's Auxiliary Memory.

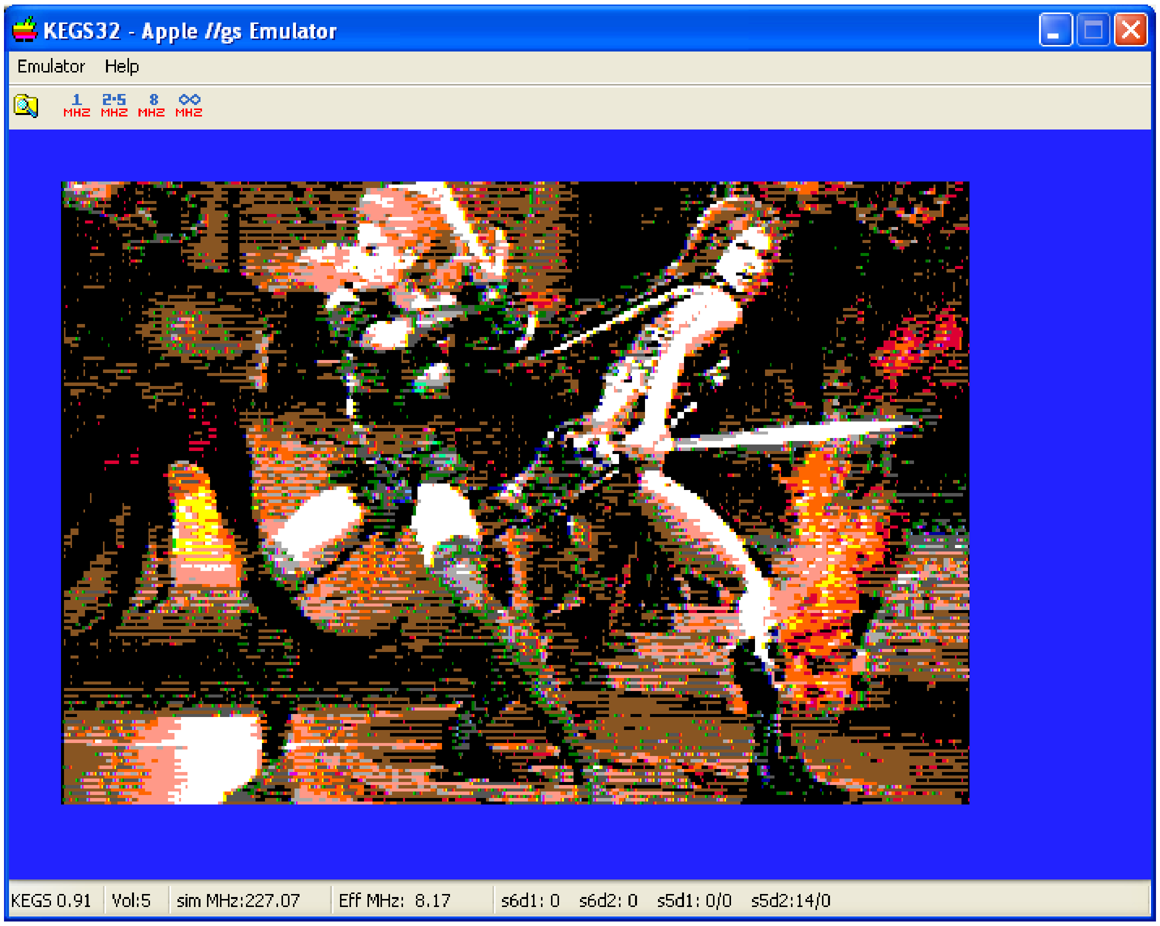
If dhishow wasn't a demo and only used the DHGR screen at $2000 and did not produce a variation that uses the DHGR screen at $4000 we could have used the better soft-switch technique used in the dloshow demo to read DHGR files to Auxiliary Memory.

This soft-switch technique is described in more detail in the following documents:

Displaying Apple II Double Hi-Res Pixel Graphics in a cc65 C Program

Displaying Apple II Double Lo-Res Bitmapped Graphics in a cc65 C Program

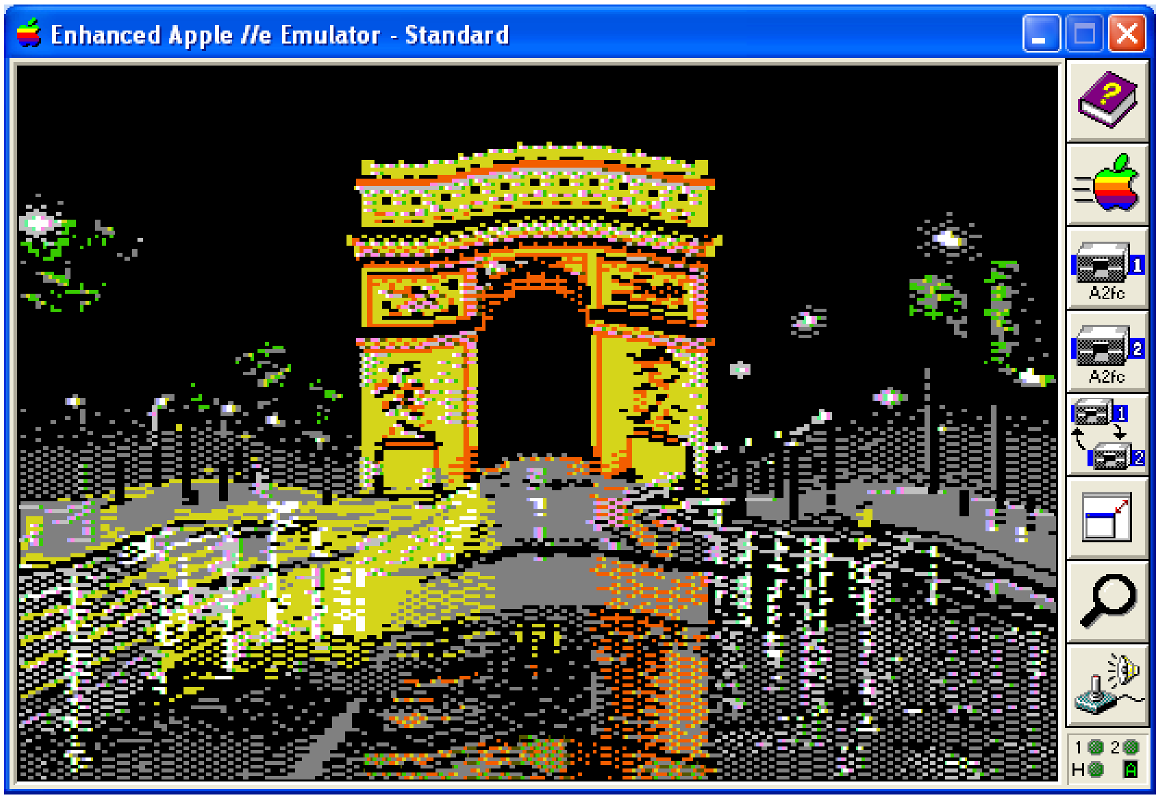
File Read and Display

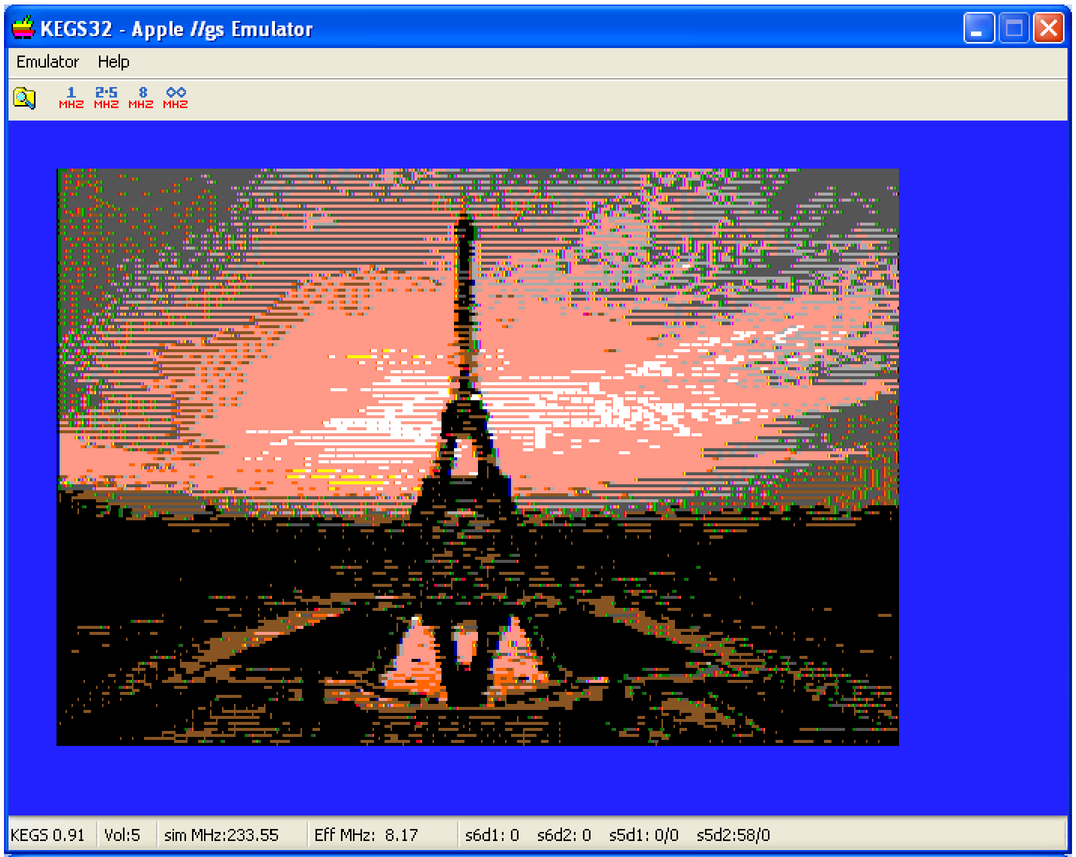

KEGS32 - Apple //gs Emulator
Emulator Help
1 MHz 2.5 MHz 8 MHz ∞ MHz
KEGS 0.91 Vol:5 sim MHz:227.07 Eff MHz: 8.17 s6d1: 0 s6d2: 0 s5d1: 0/0 s5d2:14/0

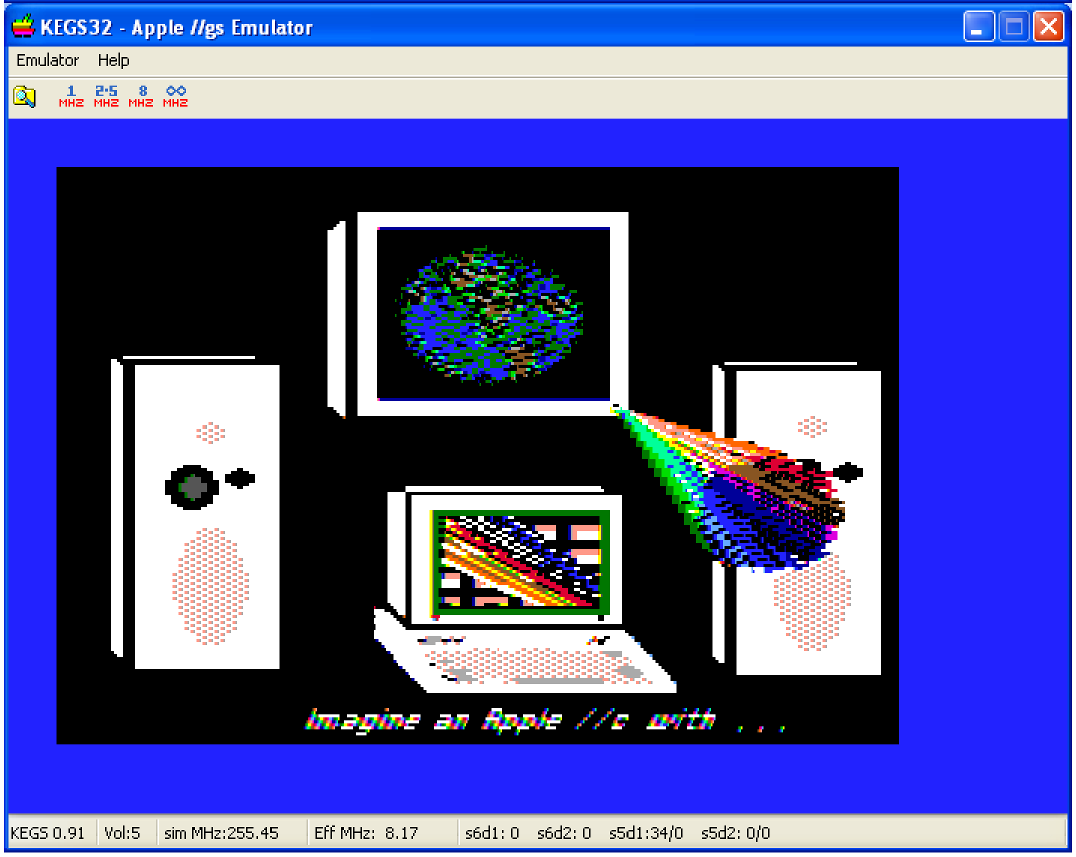
```
char lodebuf[1024];
/* The following loads two variations of BSaved
   non-compressed double hi-res graphics image formats:

   BSaved AUX, BIN Image Format file pairs
   A2FC combined file of BSaved AUX, BIN file pairs

*/
int dlodehi(char *name)
{
   FILE *fp;
   int c, fa = 0, fl = 8192, jdx, idx, status=0;
   char name2[66];
   unsigned src0, src1, dest;
   jdx = 999;
   for (idx = 0; name[idx] != 0; idx++) {
        if (name[idx] < (char)33 || name[idx] > (char)122)
            name[idx] = (char)0;
        name2[idx] = name[idx];
        if (name[idx] == '.') jdx = idx;
   }
```

```
name2[idx] = 0;

/* try to open a BASIC AUX, BIN image pair */
if (jdx != 999) name2[jdx] = 0;
strcat(name2,".AUX");

/* is it a bsaved AUX file ? */
fl = 8192;
fp = fopen(name2,"rb");
if (fp == NULL) {
    /* if not assume it's an A2FC file. */
    fl = 16384;
    fp = fopen(name,"rb");
    if (fp == NULL)return -1;
}

for (;;) {
    src0 = (unsigned)&lodebuf[0];
    src1 = src0 + 1023;
    dest = DHGR_SCREEN_ADDRESS;

    for (idx = 0;idx < 8; idx++) {
        /* read to main memory */
         c = fread(lodebuf,1,1024,fp);
         if (idx == 7 && fl == 8192) {
             /* AUX files can be 8192 or 8184 bytes */
             if (c < 1016) {
                 status = -2; break;
             }
         }
         else {
             if (c != 1024) {
                 status = -2; break;
             }
         }
         /* move to auxiliary screen memory */
         maintoaux(src0,src1,dest);
         dest += 1024;
    }

    if (status !=0) {
         fclose(fp);
         break;
    }

    if (fl == 8192) {
         fclose(fp);
         fp = fopen(name,"rb"); /* open a binary file */
         if (fp == NULL)return -1;
    }

    dest = DHGR_SCREEN_ADDRESS;
    for (idx = 0;idx < 8; idx++) {
        /* read to main memory */
         c = fread(lodebuf,1,1024,fp);
         if (idx == 7) {
             /* BIN files can be 8192 or 8184 bytes */
```

```
                if (c < 1016) {
                    status = -2; break;

                }
            }
            else {
                if (c != 1024) {
                    status = -2; break;
                }
            }
            /* move to main screen memory */
            memcpy((char *)dest,(char *)&lodebuf[0],1024);
            dest += 1024;
        }

        break;
    }

    fclose(fp);

    return status;
}
```




KEGS32 - Apple //gs Emulator
Emulator Help
1 MHz 2.5 MHz 8 MHz ∞ MHz
KEGS 0.91 Vol:5 sim MHz:233.55 Eff MHz: 8.17 s6d1: 0 s6d2: 0 s5d1: 0/0 s5d2:58/0


KEGS32 - Apple //gs Emulator
Emulator Help
1 MHz 2.5 MHz 8 MHz ∞ MHz
Imagine an Apple //c with . . .
KEGS 0.91 Vol:5 sim MHz:255.45 Eff MHz: 8.17 s6d1: 0 s6d2: 0 s5d1:34/0 s5d2: 0/0

The Main Program

As previously noted, dhishow allows a delay over-ride in a one-line text file in the slide-show working directory. When the program starts it checks for “delay.txt” and if found uses the delay value in the file instead of the default delay of 5 seconds between slides. The delay is entered in seconds but is based on a 1 MHZ machine, so if dhishow is running on an 8 MHZ machine, then a delay value of 8 is approximately 1 second.

```
int main(void)
{
    FILE *fp;
    int status = 0;
    char picname[66], c;
    unsigned delay = 5, cnt = 0;

    /* allow delay change from file if found */
    if((fp=fopen("delay.txt","r"))!=NULL) {
       if (fgets(picname,66,fp) != NULL)
           delay = (unsigned)atoi(picname);
       fclose(fp);
       printf("Delay Over-ride %d. Press any key...\n",delay);
       cgetc();
    }
```

After checking for a delay over-ride, dhishow tries to open an existing PICLIST and if not found tries to create one by making a list of DHGR files in the working directory. If that fails, perhaps because the disk is full, or there are no DHGR files in the AUX, BIN or A2FC formats that dhishow expects, then the program exits.

```
    /* try to create a piclist if it does not exist */
    if((fp=fopen("PICLIST","r"))==NULL) {

        fp = makepiclist();
        if (fp == NULL) {
            printf("Can't open PICLIST! Press any key...");
            cgetc();
            return 0;
        }
    }
```

As previously noted cc65 uses the language card memory so we need to use cc65’s library function videomode() to set to 80 column text before setting to double-res mode.

```
    /* initialize 80 column card */
    videomode(VIDEOMODE_80COL);
    clrscr();

    /* turn-on double hi-res and clear the dhgr screen */
    dhireson();
    dhiresclear();
```

The slideshow does not run in reverse or use a mouse or provide any really fancy features. If at least one DHGR file is present in the working directory, dhishow runs forever or until the ESC key is pressed to exit. As previously noted, the PICLIST is created automatically if it does not exist. If an old PICLIST has been left in place and some slides are missing, or the PICLIST has typos, missing slides will be skipped, and if all the slides are missing, dhishow will try to create a new PICLIST, and if that fails it will exit. See the notes below the next code block for when even that will not work.

An updated PICLIST can also always be auto-generated by deleting the old one from the working directory, or a differently ordered or more selective PICLIST can be created or edited in Apple, "unix", or Windows text formats. The fgets() function handles both Apple and "unix" text. The loop below skips over blank lines and lines that begin with characters that are not part of a ProDOS file name. So Windows text lines, with their extra linefeed characters just get tossed away during the process of reading the PICLIST, which allows this demo to work with a PICLIST in the 3 common text file formats.

```
/* default */
/* display each dhgr pic for 5 seconds (default) or advance on
   a key press. If ESC is pressed, exit */

for (;;) {
    while (fgets(picname,66,fp) != NULL) {
        /* load picture */
        /* skip line-feeds in MS-DOS text */
        /* skip comment lines and blank lines if any */
        if (picname[0] < (char)47) continue;
        if (dlodehi(picname) < 0) continue;
        if (cnt++ > 100) cnt = 1;

        if (delay == 0) {
            /* if delay == 0 advance on keypress only */
            /* if ESC is pressed then quit */
            c = cgetc();
            if (c == 27) {
                status = -1; break;
            }
        }
        /* flush keyboard buffer */
        while (kbhit())cgetc();

        if (delay == 0) continue;

        /* advance on timeout or advance on keypress */
        /* if ESC is pressed then quit */
        if (wait(delay)==27) {
            status = -1;
            break;
        }

    }
    fclose(fp);
    if (status != 0) break;
```

If we have gone through the PICLIST without successfully loading a single image we still can’t assume that no DHGR images exist. The PICLIST could be outdated or have garbage in it. But that logic breaks if the directory contains only HGR files and no DHGR files, or if we don’t have READ access “privileges”, and then this can loop forever with a black screen and no way to exit. That being said, I will leave it to the reader to make any improvements to this demo that may be desired but for my simple purposes I have left this as-is.

```
        if (cnt < 1) {
            fp = makepiclist();
            if (fp == NULL) break;
            continue;
        }

        /* no rewind in cc65, so we must
           close and re-open the piclist when we
           reach the end */
        if((fp=fopen("PICLIST","r"))==NULL) break;
    }

    dhiresclear();
    dhiresoff();
    videomode(VIDEOMODE_80COL);
    clrscr();

    if (status == 0 && cnt == 0) {
        printf("Can't create PICLIST! Press any key...");
        cgetc();
    }

    return 0;
}
```

Building the Demo and Links

The cc65 main page is now at: http://cc65.github.io/cc65/

Existing users of cc65 may already know what to do with this if anything, but for new users, build instructions are included below. For new cc65 users other than Windows users most of this is may only be useful for reference. Try posting a message on usenet in the comp.sys.apple2.programmer usenet news group if you need additional help setting-up your cc65 working environment on the OS that you have chosen, and if you have exhausted all reasonable alternatives including the links from the cc65 main page and googling for clues. But it is beyond the scope of this demo to provide support for setting up a news reader or for help with OS's other than Windows. All users can email still email me for help if they have exhausted the other resources that I have suggested and I will try to help on a best effort basis.

bbuckels@mts.net

Make File

The Make File for this demo works with the MinGW make as well as other makes.

```
PRG=dhishow
PRG2=dhishow2

$(PRG).PRG: $(PRG).c
  cl65 -O -t apple2enh -C $(PRG).cfg $(PRG).c
  del $(PRG).o
  cl65 -D DHGR_SCREEN_ADDRESS=0x4000 -O -t apple2enh -C $(PRG2).cfg -o $(PRG2) $(PRG).c
  del $(PRG).o
```

Windows users will need to get a make if they don't have one. Cc65 doesn't come with a make. Consider installing MinGW:

http://www.mingw.org/category/wiki/download

Or GNU Make:

http://gnuwin32.sourceforge.net/packages/make.htm

cc65 SnapShot

To build all this, Windows users can get the latest cc65 snapshot:

http://cc65.github.io/cc65/download/cc65-snapshot-win32.zip

And then create a path similar to the following (do not use spaces in the path name):

```
C:\cc65_snap\PROGRAMS\DHGR\
```

And Windows users can set-up a batch-file similar to the following:

```
C:\cc65_snap\bin\cc65-env.cmd

@echo off
SET CROOT=C:\cc65_snap
SET PATH=%CROOT%\bin;%CROOT%\tools;%PATH%
SET CA65_INC=%CROOT%\asminc
SET CC65_INC=%CROOT%\include
SET LD65_CFG=%CROOT%\cfg
SET LD65_LIB=%CROOT%\lib
SET LD65_OBJ=%CROOT%\obj
```

And Windows users can then use the Windows shortcut provided in the DHISHOW distribution at the following link:

http://www.appleoldies.ca/cc65/programs/dhgr/dhishow.zip

Unzip to: `C:\cc65_snap\PROGRAMS\DHGR\` (as noted above) or similar. Edit properties to suit and copy and re-use for other cc65 projects as required.

Linker Configuration Files

As shown in the MAKEFILE above, two linker configs are used with this demo. They are both slightly modified versions of the standard cc65 linker configs that come with the current snapshot and which are really only meant for a starting point for some of us. For a full listing of these modified configs download dhishow. It comes with source of course.

The linker config for the dhishow binary that loads at $4000 is the same as apple2enh-system.cfg with a one line change:

```
MEMORY {
  ZP:   define = yes, start = $0080,size = $001A;
  RAM:  file = %O,start = $4000,size = $7F00 - __STACKSIZE__;
  MOVE: file = %O,define = yes,start = $0000,size = $FFFF;
  LC:   define = yes,start = __LCADDR__,size = __LCSIZE__;
}
```

This leaves the screen memory at $2000 available and the memory below $2000 available, although neither area is reserved.

The linker config for the dhishow2 binary that loads at $0803 is the same as apple2enh.cfg with the 4 byte header removed. That is the only change. This header is not needed in CiderPress unless for DOS 3.3 BIN programs. I have refrained from appending CiderPress file attribute preservation tags to output filenames in my MAKEFILE to be inclusive of environments other than Windows, but that's a better idea than a header.

Andy McFadden's CiderPress can be downloaded from the following link:

http://ciderpress.sourceforge.net/

Apple Commander can be downloaded from the following link:

http://applecommander.sourceforge.net/

Working Demos

Working demos of DHISHOW are available by separate download from the following link:

http://www.appleoldies.ca/cc65/programs/dhgr/dhishowdemo.zip

They include an HDV Hard Drive Image for the AppleWin IIe Emulator which is set-up to run at 1 MHZ, and a smaller 2mg Disk Image for the kegs32 IIgs emulator which is set-up to run at 8 MHZ. Speed adjustments can be made to these by using DELAY.TXT discussed earlier in this document.

Emulators

The AppleWin Emulator can be downloaded from the following link:

http://applewin.berlios.de/

The kegs32 emulator can be downloaded from the following link:

http://kegs.sourceforge.net/

GSport is available here:

http://gsport.sourceforge.net/

For Mac Users, AppleWin will run in Wine Bottler if you make the effort:

https://groups.google.com/d/topic/comp.sys.apple2/UxMmfJ-YJe4

As will CiderPress to some degree:

https://groups.google.com/d/topic/comp.sys.apple2/0GfsM4PgqMY

But further discussion of all of this and especially the use of emulators, especially for OS's other than Windows is beyond the scope of this document and this demo.

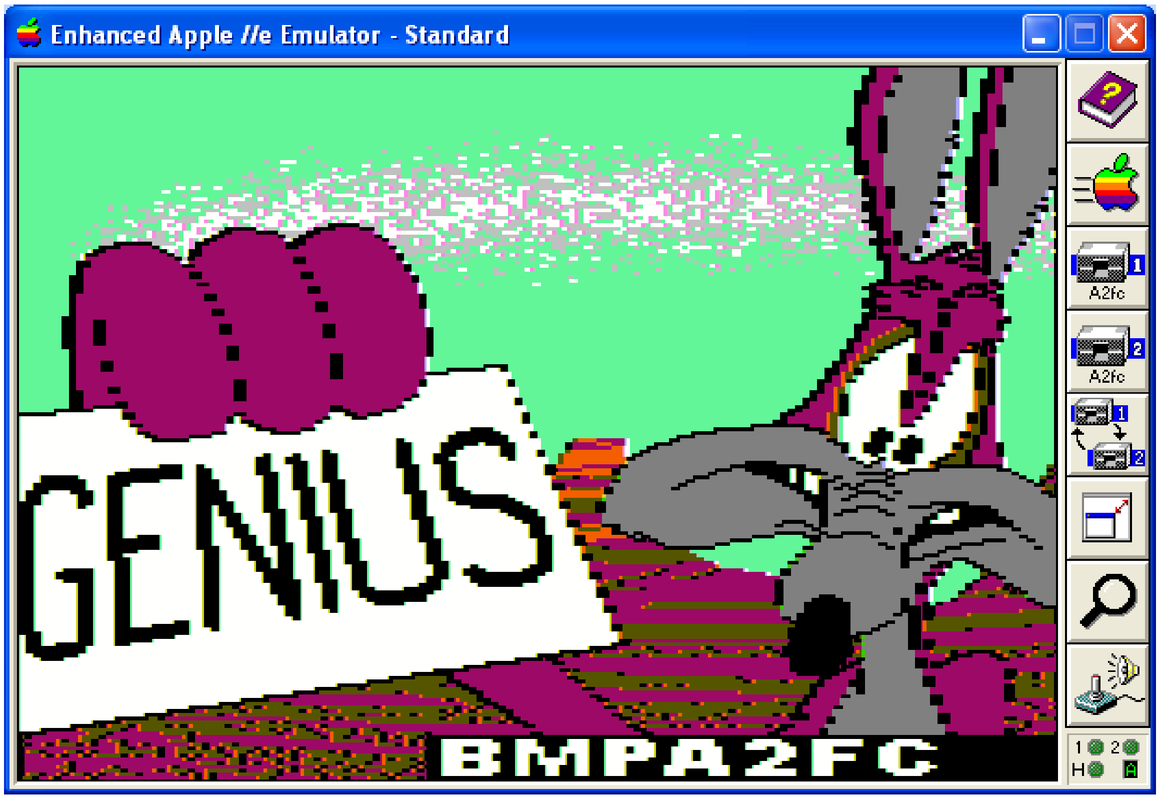
References

This demo was written almost in its entirety using the following reference:

http://apple2.boldt.ca/?page=til/tn.aiie.003

But with considerable background and help from many people.

Bill Buckels
bbuckels@mts.net


Enhanced Apple //e Emulator - Standard
GENIUS
BMPA2FC
A2fc
A2fc