Apple2000

COLLABORATORS

	TITLE : Apple2000	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		June 16, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Apple2000

1.1 Apple 2000

```
                                        APPLE 2000
The premier Apple ][ emulator for the Amiga
                by
  Kevin Kralian
                          Copyright © 1994, Patents Pending
              All Rights Reserved
```

1.2 Introduction

This program is freely distributable, as long as this instruction file is
kept with the program, and no modifications are made to my program or
instructions.

This program is also SHAREWARE (well, more accurately, Tech-Ware).
Payment is not mandatory, however, donating to me useful, enabling
technical material will result in me creating other emulations...
click here for more info

Standard Disclaimer: This program is AS IS; use it at your own risk! I
assume no responsibility if this program or its use should cause something
disastrous to happen or kill you.

I may be contacted at "Kevin_Kralian@sacbbx.com"

This program uses "ReqTools.library", Copyright © by Nico François.

1.3 Requirements

REQUIREMENTS:

- Amiga computer with Kickstart 2.0 or newer (3.0+ still untested)
- A 68020+ CPU. Emulation WILL NOT WORK on a 68000 system at this time.
- About 900k free RAM (preferably most of it FAST RAM)
- ReqTools.library by Nico François
- Apple][ROM image & disk controller ROM image
  (called _APPLE.ROM and _DISK.ROM)

Recommended:

- A two-button joystick (to emulate the Apple's two-button joystick)
- A 68030 at ~25MHz (for full speed 1 MHz emulation)

1.4 Description

DESCRIPTION:

"Apple 2000" is the premier Apple][emulator for the Amiga computer. At its current level it accurately emulates a 64K Apple][+, including:

- o 6502 CPU
- o ALL video modes (Text, LoRes, HiRes, Mixed modes, etc)
- o 16k RAM card (64k computer)
- o 5¼" disk drive (via disk images)
- o Two button joystick
- o Keyboard
- o Sound

The emulation also runs in a completely system friendly manner, multitasking properly with other programs. The two main goals were speed and accuracy. This was accomplished by hand coding the emulator in 100% machine language, optimization via instruction cycle analysis, and painstaking attention to Apple hardware details.

I feel confident that this is the fastest, most complete Apple][emulator available for the Amiga computer (commercial, public domain, or otherwise). Some of the highlights of my emulation:

- o Apple 2000 video emulation is the most accurate around:
  - There is no "dithering" of the 16 Lo-Res colors.
  - The text supports inverse and flashing characters.
  - Two consecutive color pixels are drawn as white (as the Apple does).
  - There are no missing, skipped, or fat vertical lines on Hi-Res gfx.
- o Apple 2000 disk drive emulation supports loading of "DDD" Apple disk archives from any Amiga device (no home made archive format or conversions required).
- o Apple 2000 disk drive emulation also saves disk images in compressed "DDD" format automatically.
- o Apple 2000 emulation is able to instantly load and run Apple executable files from any Amiga device (better than a real Apple; no 'disk booting' required!).

1.5 Why An Apple][Emulator?

REASONS/WHY AN APPLE][EMULATOR?

Good question! This is my first attempt at a large scale 68000 assembly program and I had nothing else in particular to write. Of course, I could always have tried to write a game or demo, however the point of this project was to learn 680x0 and more about the Amiga Operating Aystem. I did not want to delve into the hardware specifics of the Amiga (yet), such as taking over the copper, blitter, etc. An Apple][emulator was the perfect task.

Why the Apple][? Sentimental reasons. It's the computer I grew up with and learned to program on. Since I have a fundamental understanding of the Apple and because there aren't any other useable Apple emulators out

there (I've seen 5 or 6), the task called to me. I wanted to be able to play all of my favorite games that I grew up with. Yes, they certainly are not cutting edge as far as the graphics and sound goes, but they certainly are playable! And I can overlook the cosmetics for some good gameplay (i.e, just like people appreciate classic cars or oldies music). Plus I wanted all of my friends to be able to play all of those great forgotten games...the classics! The original CASTLE WOLFENSTEIN, CHOPLIFTER, KARATEKA and CARMEN SANDIEGO. How many other multitasking versions of JUNGLE HUNT or ROBOTRON 2084 can you play while downloading a program? By writing this one emulator, the entire Amiga community is suddenly presented with over 10,000 (now multitasking) Apple][programs we wouldn't have otherwise been able to use (or play).

After letting the idea stew in my head for 6 months, and much apparent rambling to my friends (who so nicely encouraged me by saying, "What? YOU write an emulator? And in C? UGH!"), I began coding. One month later, I brought my first creation over to a friends house to see how it worked on his system. After starting it up, we sat there. 30 seconds later we were still sitting there, looking at a white screen. Eventually, we watched as each little white character s-l-o-w-l-y was replaced by a black space. Two minutes later, after getting bored of waiting for it to finish clearing the screen, we gave up and played 2-player LEMMINGS. I knew the only way I was going to be able to make this program 'practical' was to do it in assembly.

I finally bought DevPac 3. After writing a program to bounce 65,535 colored pixels around a screen, I felt ready and experienced. I began converting my routines for my emulator into assembly code. Almost two years later (and after rewriting most of my emulation 10 times) my emulator has finally matured enough to go out into the cold and brutal world. Here it is, ready to be challenged by thousands of Apple programs I have never even heard of, and ready to do its damndest to run them all!

1.6 Running the Emulation

RUNNING THE EMULATION

Make sure "ReqTools.library" is in your libs: directory and place "Apple2000", "_APPLE.ROM", and "_DISK.ROM" all in the same directory. Then from the CLI/Shell, CD to its directory and type "Apple2000".

Now, assuming a little common sense (press the "OK" button on the window!), you will see a black screen with "Apple 2000" in a title bar at the top, and the words "Apple][" immediately below it. Congratulations, you are now using an Apple][computer. The Apple is trying to boot a disk.

I will assume you have a little knowledge on using an Apple][. Here are some of the pertinant keys:

KEY	Function
DEL	Apple "Reset" key.
ctrl-DEL	Similar to "Ctrl-Open Apple-Reset" on][e,][c,][gs. Forces reboot, even if reset vectors have been changed.

```
 RAmiga-Q          Quit the emulator (after verification).
 RAmiga-L          Load Apple disk image or executable into the emulator.
 RAmiga-S          Save Apple disk image.
 L-ALT             (Like Open-Apple on ][e) Represents Apple Paddle
                   Button #0
 R-ALT             (Like Closed-Apple on ][e) Represents Apple Paddle
                   Button #1
                   (Alt keys do not affect other keystrokes to emulator)
    ^
    |              Arrow keys patched to be like Apple ][e, ][c, ][gs.
 <--+-->               (Note: Apple ][+ had no Up/Down arrows, and most
    |                   older programs won't handle them as expected.)
    v

 NumPad only:
    8              Trim Apple joystick center position in respective
  4 5 6            directions.
    2              "5" will reset it to default (of 127,127).
```

1.7 Loading/Saving Disks/Files

LOADING DISKS/FILES

Once the Apple is running, you'll probably want to load an Apple disk or executable. Here's how: at ANY time during emulation, feel free to press Right-Amiga-L to bring up the Load File requester. From this requester you may load Apple 5¼" disk images or executable files. Simply navigate to wherever the files are kept and load the file/disk image you want. Apple 2000 recognizes several types of load files:

- Filenames with a <xxx> suffix are Dalton Disk Disintegrator archives (DDD was a common disk compression util for the Apple, similar to DMS for the Amiga) and the emulator will automatically decompress them!

- Filenames with a .DISK suffix are raw disk images with no compression. They are capable of storing images of non-DOS and copy-protected disks, but are about 220k in size. This format is primarily due to early testing of the emulator and will be phased out (as well as any references to it).

- Filenames with a .PROG suffix are executable files; these are single files that were runnable from Apple DOS 3.3/ProDos and did not require any disk access thereafter. These files now do not even require booting any Apple disk and are simply loaded into the appropriate Apple memory areas and started instantly (quicker and easier than a real Apple!).

After loading a disk image (compressed or not), the emulator will ask you if you want to 'boot' the disk. If you choose not to, you have effectively just 'put the disk in the drive' (useful when you need to insert 'Disk 2'). On the other hand, loading an executable Apple file does not give you any choices and immediately runs it. This has all been designed to keep the emulator as clean and simple as possible in terms of starting and running Apple programs for the non Apple-literate user.

Keep in mind, loading a disk image is the same thing as inserting the disk into the Apple drive. It will STAY there until you replace it with another disk (or some program erases that disk). Even after you load and run several executable Apple programs, hitting Ctrl-DEL (rebooting the Apple) will boot up the last DISK IMAGE you loaded (if any). This can be confusing if you don't know whats going on (i.e, after finishing playing MS. PACMAN and reseting the Apple, why is MUSIC CONSTRUCTION SET loading? Because the disk is still in the drive from before).

SAVING DISK IMAGES

Pressing Right-Amiga-S will bring up a requester to save a disk image. Disk images are automatically compressed when saved. Navigate your way through the directories (as you normally would), and enter a filename (like normal). The only rule here is that the filename MUST end with a > character (greater than sign). This is because the original DDD archives did not have any special identifying header, but merely relied on that trailing character (actually, DDD uses <###> where the #'s represent the number of sectors the file takes, but my emulation ignores that value and just checks for that last character).

NOTE: Ultimately, these archives will be saved with an Apple ProDOS header so that files can be transferred right back to real Apples and decompressed. However, due to a lack of resources, I have not yet been able to implement this 'identical header' and currently disk-images will not transfer back to an Apple. This will be incorporated later.

1.8 Transferring Apple Files/Disks/ROMs

TRANSFERRING APPLE FILES

To get an executable binary file from a real Apple to the Amiga is quite simple. Copy the file to a ProDOS disk (using a utility such as Copy][+) and then transfer it via null-modem (or however you want) to the Amiga. Make sure to append a .PROG to the end of the file name so that it's recognized by the emulator.

NOTE: At the moment, I require the conversion to ProDOS simply because ProDOS puts a standard header in front of the file that my emulator needs. I'll eventually allow files to be transferred directly from DOS 3.3.

TRANSFERRING APPLE DISKS

Only UNPROTECTED standard 16 sector Apple disks are currently useable. This eliminates copy protected software. Simply run Dalton Disk Disintegrator (do not use version 2.0! It has a bug! Use version 2.1) and use it to compress the disk into a file. Then, convert it to ProDOS and transfer it to an Amiga (as described in "Transferring Apple Files" above). Once on the Amiga, change the name to end in <xxx> because ProDOS strips out the <> characters.

NOTE: Like with files, eventually this will not require the intermediate

ProDOS conversion either.

TRANSFERRING APPLE ROMS

The Apple emulator, being true to form, requires the actual Apple ROM data in order for the Apple to do anything. The standard Apple ROMs in use were the 'AppleSoft ROMs that contained AppleSoft BASIC, the assembly language monitor, and autobooting code. So I suggest that you obtain the same ROM if you would like the same compatibility. The ROM image can be obtained by booting an Apple][or][+ with DOS 3.3, then typing:

BSAVE BASICROM,A$D000,L$2FFF

to save it to disk. Also, the disk controller Rom is saved by typing:

BSAVE DISKROM,A$C600,L$00FF

Incidentally, the main ROM image is on Apple's "DOS 3.3 System Master" disk, called FPBASIC (which may be used instead). After saving these images to disk, use your favorite terminal software and a null modem cable (or real modems or whatever you like) to transfer these files to the Amiga. Once transferred to the Amiga, give these files the proper names and place them in the same directory as the Apple2000 executable.

Theoretically, you can use the ROMs obtained from an Apple][clone (i.e, Franklin Ace, PineApple, etc.), but keep in mind that these ROMs were not 100% compatible (but were quite close). This would effectively make my emulation a "Franklin Ace Emulator". :-) However, you CANNOT use the ROM images from an Apple][e,][c, or][gs (maybe eventually...we'll see)!

Once on the Amiga, the filenames MUST be _APPLE.ROM and _DISK.ROM

1.9 Paddle/Joystick Emulation

PADDLE/JOYSTICK EMULATION

The Apple][commonly uses either 2 paddles, a joystick, or a graphics tablet (like a free-floating joystick). My emulation covers all bases. The paddles and graphics tablet emulation are handled with the mouse and the joystick emulation is handled with the joystick.

The F9 key toggles between the two and displays your choice at the top of the screen.

For the Joystick:
The Apple][uses (unfortunately) the variable 'all-over-the-place' PC-type of joystick (with two buttons). This evolved from the earlier Apple days when paddles where common (PADDLE - a turning knob with a button, useful for playing PONG). Anybody having used these joysticks know the frustration of having to constantly 'center' or 'trim' the joystick for different programs. The emulation takes this into account also.

The Amiga 'Atari-style' joystick in the normal joystick port is used (preferably, a true 2-button joystick). The center position defaults to the optimal center positions on an Apple (127 X 127). However, different games expect different centering values, which can be trimmed with the 2, 4, 6, & 8 keys on the numeric keypad (ONLY, see
Running Emulation
).
For example, if you start CHOPLIFTER and your character drifts towards the left, press the "6" key to center the joystick more towards the right until your character no longer drifts!

If you do not have a two button joystick, you have two choices. Go buy one, or else just use the Right-ALT key in lieu of the second button (by the way, the two ALT keys work great for pinball games like RASTER BLASTER).

For the Paddles/Graphics Tablet:
Some of the older Apple games were designed to be used with paddles, not joysticks. This is noticable in games (APPLE GALAXIAN) as in when you release the joystick, your ship automatically moves back towards the center point on the screen. This is exactly what would happen on a real Apple with a joystick. You need to use the paddle emulation here. Push F9, the top of the screen should say "Mouse", and now the mouse controls this stuff. The horizontal movement controls Paddle #0, and vertical movement controls Paddle #1. If you have a game that needs independant control of each paddle, you are out of luck (I'll get to that later). Programs designed for graphics tablets (KOALA PAD) or un-centered joysticks work great in this mode also, i.e, FANTAVISION, MISSILE COMMAND games, and most other free floating cursor control programs work ideally. The mouse works just like you would expect here.

1.10 Tech Notes

TECH NOTES

Though the emulation is 100% system friendly and running in a standard intuition screen, it has to do a few tricks with copperlists in order to achieve the Apple mixed screen graphics. The side effect of this technique is that when you pull down screens in the foreground to reveal the Apple emulator in the background, there will be times that the display looks garbled and/or is flashing. This is normal...nothing is wrong. If you don't like it, then don't pull screens down in front of the emulation! (A man tells a doctor, "It hurts when I do this...", and the doctor tells him "Then don't do this.")

My emulation uses innovative 6502 emulation routines, which are significantly faster than any other 6502 emulations that are available on the Amiga (commercial or otherwise). My Apple emulation also has the most accurate graphics emulation I have witnessed. As complex as this is, speed is always maintained by extensive use of complex lookup tables (several hundred K worth).

The ONLY graphics glitch is that the Hi-Res graphics screens do not fill

in the entire display width. That is, they leave a half-inch black border on each side of the display. Why? Because the Apple Hi-Res screen has a horizontal resolution of 280 pixels, and the Amiga's display has a minimum resolution of 320. Trying to stretch this display by leaving an empty pixel after every 7 pixels or drawing every 7th pixel twice, results in a highly distorted and uneven image. The Text modes and LoRes modes still use the entire screen width (to maintain aspect ratio). This slightly narrow display is only noticeable in the 'mixed Graphics/Text' mode, where text will be a little wider than the graphics above it.

Also regarding Text and Graphics (but not a glitch, it's an improvement) is the fact that mixed Graphics and Text on the old Apple]['s used to cause the text to be fringed with green and purple instead of being solid white. This fringing has absolutely no purpose, but is a mere artifact of the Apple video circuity. My emulation cleans it up with crisp & clean text output all the time (does anybody have any complaints?). Apple finally cleaned this up with the Apple][gs and its RGB output (but introduced a couple other graphic glitches), so I believe my clean Text display is desirable.

Some Apple programs use "unimplemented" 6502 instructions. These are instructions that are not official, but partially decode into doing a particular function (as discovered by many unorthodox programmers). My emulation does not support ANY unimplemented instructions, and will simply break upon hitting any of those instructions (with Apple][software, I have seen very few programs that do this).

1.11 Planned Improvements

PLANNED IMPROVEMENTS

Currently, emulation speed is pretty much as fast as possible under the current 'interpreted' method. A speedup of about 3X is possible if I do 'pre-interpretation' which essentially converts 6502 code to native 680x0 code ahead of time, then running it at full speed. However, this comes at a cost of excessive memory usage (I estimate using about 2 megs for the 64k Apple emulation). This could be considered if enough people are interested (this would be the final speed boost required for the A1200 owners if they have enough memory!), but is of low priority for now.

A 68000 version is easily possible, but emulation is so slow at that point (games are frustratingly unplayable), that I haven't bothered to do one.

I plan to add Apple printer & serial emulation (and redirection). This way you could redirect Apple printer output to an Amiga file or to an Epson emulator (to print Epson output to any Amiga Prefs printer). Or emulate the Apple serial card with an Amiga Modem, etc.

I plan to (eventually) upgrade the entire emulation to Apple][e /][c status. This includes Apple "Double-Hi-Res" graphics, 128k RAM, and 80-column text.

I have thought about writing a ProDOS driver allowing the Apple to access Amiga devices as an Apple hard drive (is anybody using the emulator this seriously?).

I might tackle using a real 5¼" disk drive if enough people want it.

1.12 What About EMPLANT?

WHAT ABOUT EMPLANT?

My emulator, "Apple 2000" was (p)reviewed in Amiga Computing (Issue 71, March 94), inside a larger review for the Emplant card (there's even a screenshot where you can read my title bar, Apple 2000!). For all intents and purposes, the review makes it appear as if this program was written by, owned by, and coming soon from Utilities Unlimited, makers of the Emplant card (regardless, the reviewer loved it, noting that this was the fastest 6502 emulation he has seen).

At several World Of Commodore shows, Jim Drew showed my early versions of "Apple 2000" to crowds of people during his presentations of his Emplant card. A friend even has a video-tape of Jim loading up and showing my emulator to a crowd when I asked, "What other emulators are you doing?" (before he knew who I was) at WOC in Pasadena, 1993.

To set the record straight, I did send Utilities Unlimited several early exclusive 'evaluation' versions of my emulator to see if they were interested in purchasing it (for their Emplant package). But this program (Apple 2000) is not a part of the Emplant package! Utilities Unlimited was in no way involved with the development of this program. There was never an agreement made over ownership rights. Several proposals were submitted by me as per Jim Drew's requests for exclusive rights, but an agreement was never reached.

The positive side effect of this is that you may use the Apple 2000 program without having to spend >$300!

1.13 About the Author

ABOUT THE AUTHOR

"Apple 2000" was written by Kevin Kralian over the course of two years. He has spent time in the US Marine Corps 'finding himself'. He is now a full time college student, preparing to transfer to CSU, Sacramento. He has over 10 years of programming experience, including ADA, BASIC, C, Pascal, 6502 and 680x0 assembly. Programming interests focus on performance programming, including games and emulation. Career goals include firefighter (!?) and computer programmer.

He may be contacted at "Kevin_Kralian@sacbbx.com"

1.14 Payment

"PAYMENT" FOR THIS PROGRAM

Though this program is being distributed freely as shareware, I do not expect monetary payment. My original intentions were simply to have my program be 'used' by the Amiga community, and I still feel the same way. I've worked long and hard on this program and the most rewarding thing to me know would be to simply know people are enjoying it!

However, what I WOULD appreciate would be any technical references for any computer/hardware/platform. Let me explain...

Many improvements in the Apple emulator are dependant upon me finding Apple technical reference material (i.e, unimplemented instructions, serial/parallel support, ProDOS harddrive support, etc). If you would like to see these features implemented, the biggest thing you can do is send me any tech material that could be helpful (i.e, "Whats Where In The Apple][: An Atlas", "Beneath Apple DOS", "Beneath Apple ProDOS", etc).

Also, some ideas for my next emulator include: Atari VCS (2600), GameBoy, Nintendo, Atari 400/800 and Commodore 64/128. Even though there are a few C-64 emulators out there, many people have urged me to do one "the right way" (ground has been broken and the C-64 emulation is underway already). I tend to want to do the old Atari VCS or Gameboy emulation. HOWEVER, in order to do this, I need the tech information that I cannot publically obtain.

Do YOU want these game machines to be emulated (I do)? If you are one of those priviledged people who might have been involved in developing software for any of these machines or somehow have any tech info on these machines, please send me any and all tech information. *** I WILL *** make an emulator of these machines when I have enough tech information to do so. But I need your help.

I am open to any suggestions, comments, or feedback. Let me know how the emulator works for you. Please let me know of anything that does not work (that works on a real Apple][), and I will do my best to correct the problem. I am also interested in obtaining any Apple][programs people may have to test under my emulation.

Anybody interested please contact me at "Kevin_Kralian@sacbbx.com"

Particular things I'm looking for:

- Whats Where in the Apple][: An Atlas to the Apple computer
- Beneath Apple DOS
- Beneath Apple ProDOS
- Apple Super Serial Card / Parallel card manuals
- AmigaDOS Programmers Reference
- Any 2.0+ Amiga ROM Kernal Manuals (I'm using 1.3)
- ANY kind of tech info on Gameboy, Nintendo, or the old Atari VCS (there once was an Apple][card to program the Atari. Anybody have it?)
- Any and all Apple][programs.
- Any old Apple][hardware (I use a][gs; its too different from][+)
- Any responses, reactions, suggestions, etc. on my emulation.

o etc...

1.15 Credits

CREDITS

I owe lots of thanks to lots of people.

Thank you my dearest JoAnnaBear for being so supportive of me and this project over the last two years, and for not going crazy over my many hours of "techno-babble", but just patiently smiling back as if you understood me. :-)

Thank you Robbie for all your inspiration and encouragement. And thanks for your ice-cream sandwiches, twisted sense of humor, brainstorm sessions, and hundreds of hours worth of second-hand smoke (cough cough). Thank you for the book "Amiga Machine Language Programming Guide" - the very first 680x0 assembly book I've seen (blech!). By the way, this book was due back at the library in 1989. How are your games "To Sir With Love" and "The Piano" coming along? Oh yeah...and thanks for cleaning up and converting my docs to AmigaGuide format for me :-)

Thank you Bill, for taking your family and moving far away (just kidding!).

Thanks to those incredible guys at Computer Cafe. I appreciate how you let me use your various machines for debugging and testing during the development of my emulation. Without your help, I would have never been able to work out the '040 bugs, nor have seen my emulation running on a 28" monitor with cool 24 bit backgrounds.

Thanks to Carmen Rizzolo, the computer artist extraordinaire! Your original artwork for my previous programs are utterly amazing. Without people like Carmen, where would we get cool 3D Star Trek and telephone objects?

Thanks to Will, the only intelligent Mac owner I know. It was great to share ideas on high performance 6502 emulation with the 680x0. Have you finished your Mac version of your Apple][emulator yet? Thanks for that 'half' of the "Inside the Apple //e" manual. Did you ever find pages 1-110?

Thank you Nico François, for your contribution to the Amiga community. ReqTools is a very polished piece of work, and I know that your work has saved me (and many others) hours of work trying to "recreate the wheel". (Reqtools.library is Copyright © by Nico François).

Thanks to the many helpful people on the Internet, for helping me through many obscure programming and debugging challenges.

Thanks to Apple Computer and Steve Wozniak for creating the original Apple][computer. And congratulations to Apple Computer for knowing how to market their computers and becoming a large, successful company. Maybe Commodore can learn a few things from you before they drive themselves out of business?

Thanks to 'Dalton', for his "Dalton's Disk Disintegrator" (DDD) program on the Apple][. My (de)compression routines were based on his routines and attempt to compress data in an identical, compatible way.

And finally, thanks to the many people I do not have space to mention, and to all of the Amiga users who have made the Amiga scene as wonderful as it is.