EMULATE!

COVER FEATURE

Dust off the greatest games ever made – then play them! Neil Bothwick, Richard Drummond, Mike Saunders and Nick Veitch go one up.

ON the DISC

CONTENTS

Coding and gameplay go together like Crack and Attack, so it's hardly surprising that Linux developers have invested valuable noodling time in bringing some of their favourite old computer games to their new adopted home. And if that happens to include the Commodore 64 version of *Attack of the Mutant Camels*, no problem. Because while Linux itself runs on practically any modern hardware, being a hotbed of developers it is also a great platform for emulating other hardware, too.

In fact, Linux is widely used by software developers to create the games in the first place, because often development starts before real hardware is available. It wouldn't surprise us at all if PS3 games were being developed on emulated hardware under Linux as we speak.

Practically every gaming platform you can remember has at least one emulator available under Linux, from the ZX80 onwards. We ran a huge series on emulation stretching from *LXF13* to *LXF33*, penned by Simon N Goodwin (now reprieved at **http://simon.mooli.org.uk/LXF**), and this covered pretty much everything, as it ran to some 80-odd pages in all. For this feature we have ignored some of the older hardware – emulators for the Acorn Atom, BBC Model B and the like haven't really changed much in the interim.

There are other reasons why you might want to use the software here to emulate old hardware. You may have a lot of ARexx scripts for *ImageFX* you want to use, or some old documents in *PageMaker* on the classic Mac. By far the most common use of emulators, though, is to recapture all those lost hours playing the simple, primitive but highly addictive games of your youth – and lose them all over again.

»

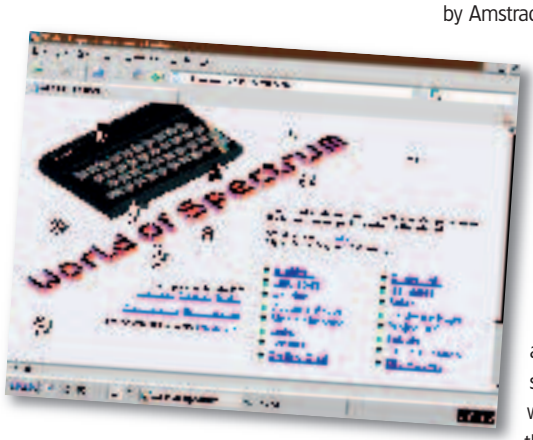
DISCLAIMER

The laws regarding emulation differ throughout the world, so we advise you to make sure that you are acting within the relevant laws of your country. As a general guide, downloading copyright ROMs and games from the internet is unlawful pretty much everywhere. You may be entitled to run emulations if you own the original hardware and/or software, but we can't give you legal advice – check out your local laws!

ZX SPECTRUM

Sorry, there's no rubber keypad emulator...

For many, Clive Sinclair's ZX Spectrum was *the* machine of the eighties. It was the system that got a lot of geeks started in the world of computing. Launched in 1982 with a comically awkward rubber keyboard, the early Spectrums were just about usable for BASIC programming and serious software; however, it was the growing games market that brought it massive sales, mainly in the UK.

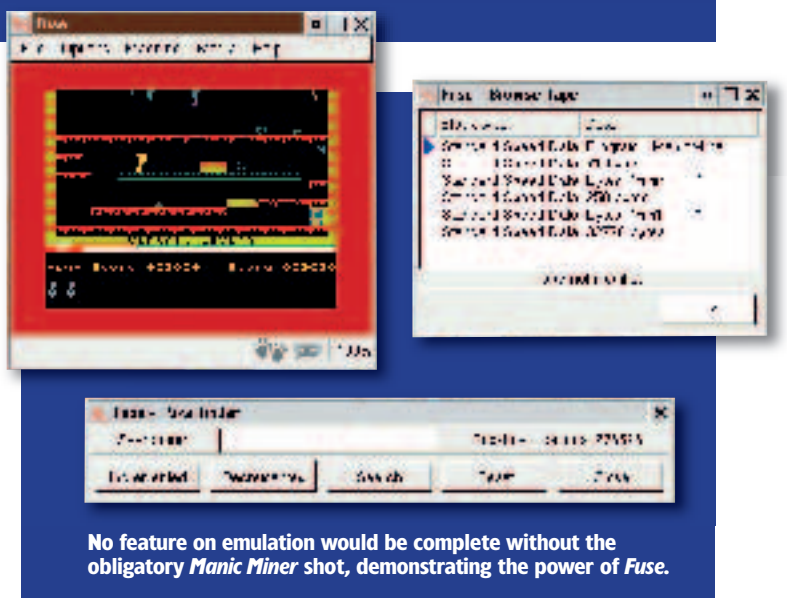


World of Spectrum is a really excellent site, jam-packed with emulation info.

SYSTEM SPEC: SPECTRUM +2A

CPU: Zilog Z80A @ 3.54 MHz
RAM: 128k
ROM: 64k (BASIC, DOS etc.)
Sound: AY-3-8912 (3 channels)
Max res: 256 x 192 pixels

Sinclair Research was bought out by Amstrad in 1986, leaving Clive to focus on curious contraptions such as ZETA (an electric motor for pedal bikes – the mind boggles). With the more typist-friendly Spectrum +2, Amstrad grew the machine into a highly successful gaming platform, with magazines and specialist stores supporting it all round. It wasn't all entertainment though: compilers, word processors and graphics tools were produced, and the machine still has a large fanbase in the retro scene today.



No feature on emulation would be complete without the obligatory *Manic Miner* shot, demonstrating the power of *Fuse*.

There are several Spectrum emulators for Linux, all in varying stages of development, but the leaders are *Fuse* and *XZX-Pro*. The latter is shareware *(see Emulator Options box below)*, so we'll get started with the former here: the *Free Unix Spectrum Emulator*.

***Fuse* in action again, showing the memory browser, model selector and general options.**

PLAYER INSTRUCTIONS

1 Install *Fuse*. If you're building from source, you'll need to compile the **libspectrum** supporting library first. This separates the file-handling routines of the emulator into a separate bundle, ideal for use in other programs. If you've got *GTK 2.x* installed (most likely on all recent desktop distros), *Fuse* will build with a shiny GUI – otherwise it'll fall back to vanilla *X* or *SDL*. We've included some binary packages on our coverdisc that'll work on most distros.

2 Start it up. Running **fuse** from the shell prompt will pop up the main window of the emulator, complete with 48K BASIC prompt. Usefully, *Fuse* can switch between other machines in the Spectrum line – go to the Machine menu and choose Select (or press F9) and a dialog will appear. Here you can opt for almost every model, from the original 16K through to the Spectrum SE. When you've selected, *Fuse* will reset the CPU and bring up the appropriate starting screen.

3 General configuration. In the Options menu, selecting General brings up an overall settings window. There's no need to alter most of the defaults, but if you're having problems with performance you can alter the emulation speed percentage and frame-rate value to get it running at a zippier pace. For sheer nostalgia overload, you can even recreate a black and white TV – ah, the memories! If you're planning to use the emulator for gaming, dive into the Joysticks submenu under Options, where you can assign keys for the controller.

4 Specific tuning. At this point *Fuse* is ready to go, but you may want to tweak a few other options beforehand. Via the Peripherals menu item you can enable various add-ons (not essential), and the Sound entry includes a few toggle-able settings for finer control.

5 Load a game! Emulated Speccy programs come in a multitude of formats, the most common being .z80 memory snapshots and .tzx/.tap cassette tape files. *Fuse* also supports .szx and .sna snapshots, together with .zxs and .snp, so it'll handle almost every format with ease. (It'll even decompress them on the fly if they're in gzip or bzip2 format.) Open a game from the File menu, and *Fuse* should automagically load it up in the emulator, ready for play.

6 Manage files. With a game or program up and running, you can save a snapshot of the machine's RAM via the File menu. This'll restore the machine to the exact same state as when you saved it. Under the Media menu there are a few options for browsing round the tape's contents, plus you can save screenshots with the File menu. For total Speccy addicts, two features stand out: the ability to record and re-watch your play via the .RZX save function (under File), which lets you send the results to other emulator users, and the AY recording option, which saves music tracks. Top notch.

7 Find out more. *Fuse*'s home is at http://fuse-emulator.sf.net, where you'll find the usual collection of development and feature news, and *XZX-Pro* can be found at www.zx-spectrum.net/xzx. One of the best sites for further information on emulators, along with stacks of games and demos, is the brilliant www.worldofspectrum.org.

EMULATOR OPTIONS: XZX-PRO

Fuse is a remarkably solid emulator, running all the games and apps we threw at it, but a special mention should go to *XZX-Pro*. This emulator is bulging with features – all kinds of hardware and peripherals coded in, a vast array of file formats supported, and there's even a built-in VNC server for multiplayer games. Gadzooks.

To pay for all this coding effort, though, the author has made *XZX-Pro* shareware. You can download a trial version (some features disabled), and then consider paying €25 for the full version, along with some extra utilities. A small price to pay for such a fully-fledged emulator.

COMMODORE 64

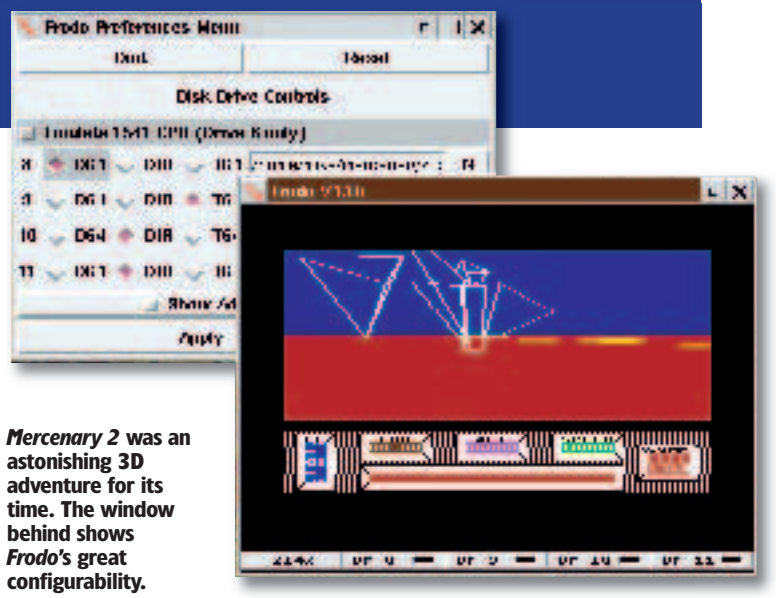
You can almost hear the tape machine creaking...

Flame wars in the free software world may be fiercely intense – KDE vs Gnome and *Vim* vs *Emacs* have left a few third-degree burns – but in the eighties you couldn't take part in the Spectrum vs C64 debate without a trip to A&E. Commodore's machine was graphically the more powerful of the two, but technical might isn't everything; heated arguments covered all aspects from software range (important) to design of the logos (erm). Released in 1982, the C64 provided stacks of computing power for a much lower price than similar machines, and continued production right up until 1993.

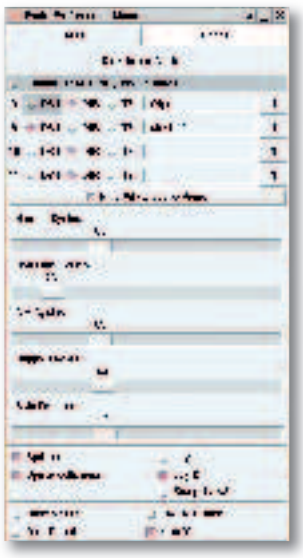
SYSTEM SPEC: COMMODORE 64

CPU: MOS 6510 @ 0.985 MHz
RAM: 64k
ROM: 20k (BASIC and Kernal)
Sound: MOS 6581/8580 (3 channels)
Max res: 320 x 200 pixels

Like the Spectrum, the C64 was primarily cassette tape-driven, and included a version of BASIC. Around 10,000 commercial programs were released for the machine during its life – a clear indication of how successful the machine was, and of why there are stacks of emulators in development today. Indeed, there's even a lively hacking community for the C64, still producing demos and hardware addons, but nothing can rival the amazing Contiki (**www.sics.se/~adam/contiki**) – a graphical, multi-tasking OS for the machine complete with internet apps.



***Mercenary 2* was an astonishing 3D adventure for its time. The window behind shows *Frodo*'s great configurability.**



***Frodo*'s interface could do with a spot of polish – it's still workable, though.**

Two of the best free C64 emulators are *Frodo* and *VICE*. The latter is more of a power user's tool, bundled with features aplenty, whereas *Frodo* is a lot easier to approach and get running. Call us wimps, but without further ado we'll step through the process of installing *Frodo*, setting options for maximum emulation goodness, and working with software. If you're tempted by *VICE*, see the *Emulator Options* box below.

***VICE* includes a bewildering array of menus for all manner of system settings.**

PLAYER INSTRUCTIONS

1 Compile *Frodo*. Once it's extracted, jump into the **src** directory and issue **./configure**. If all's well and the dependencies can be found (there's nothing out of the ordinary), a quick **make** will build the emulator binaries. Then step back into the main directory – there should be a **FrodoPC** binary ready to run, alongside a number of supporting hardware files for the OS (such as a **'kernal'**). Additionally, the **TkGui.tcl** script provides a rudimentary front-end for the emulator, so if you have Tcl and *Tk* installed, it should start up automatically when you run **./FrodoPC**. Binaries can be found on our disc.

2 Explore the GUI. The chunky *Tk* front-end isn't the most attractive thing ever designed, but it does the job. Most settings are self-explanatory, and the documentation (found in **Docs**) provides much more info. Most importantly, the boxes along the top let you drop files in as virtual tapes or disks; you can also provide a directory of programs, such as the samples supplied with *Frodo*.

3 Advanced options. In the GUI is a checkbox that opens up a bunch of detailed settings, most of which don't need to be changed unless you have specific requirements. The Skip Frames slider affects how often the screen is redrawn – this is the most effective way to speed things up if you're on a low-spec system that's suffering speed-wise. Conversely, if the emulation is running too fast, select the Limit Speed toggle just beneath; it'll constrain *Frodo* to run at 100% accuracy of the original C64.

4 Load a game! The emulator will read .d64, .x64, .t64 and LYNX files, so if you add one into the number 8 box at the top of the options screen, and select the appropriate button, you're ready to go. With a virtual disk in the drive, for instance, you can reset the machine, enter **LOAD "*",** and then **RUN** once it's done. The lights beneath the main emulator window indicate disk activity.

5 Get familiar with the keys. Ninety per cent of the keyboard matches the C64 equivalent precisely, but there are a few extras: Esc is Run/Stop, right Ctrl is the Commodore key, and F12 performs a reset. By default, the numeric keypad emulates a joystick – this is adequate for many games, but thankfully real controllers are supported too.

6 Go further. The **Docs** directory contains a wealth of info on *Frodo*'s extra features, such as the SAM debugger and Kernal additions, and its homepage is at **www.students.uni-mainz.de/bauec002/FRMain.html**. Meanwhile, the *VICE* emulator can be found at **www.viceteam.org** and there's a colossal pile of C64 emulation info at **www.c64.com** with masses of links to related sites.

EMULATOR OPTIONS: VICE

VICE (the *Versatile Commodore Emulator*) is a top-notch C64 emulator geared towards more experienced users, and has been ported to many platforms (Unix flavours, Windows, BeOS, even RISC OS). Its compatibility and feature set are first-rate, and it uses Plan *X* widgets so very few dependencies are required.

If you install *VICE* but wonder where the binaries have gone, try running **x64** to start the emulator in C64 mode. Most of *VICE*'s settings are accessed by left- and right-clicking on the main window. This brings up a maze of options and features – definitely the better choice for experienced C64 fans.

SUPER NINTENDO

Can Princess still beat the Mushroom?

***Snes9express* makes launching and tweaking *Snes9x* orders of magnitude simpler.**

***ZSNES* includes its own GUI for configuration – it's all easy to navigate.**

SYSTEM SPECS: SNES

CPU: 16-bit 65816 @ 3.58MHz
RAM: 128K, 64K video
Sound: Sony SPC700 (8 channels)
Max res: 512 x 448 pixels
Max sprites: 128

Astoundingly, Nintendo's history stretches back over a century. The Japanese company began business in 1889 as a producer of playing cards, led by a craftsman called Fusajiro Yamauchi. After taking control of the business in 1949, his great-grandson Hiroshi Yamauchi helped the company to blossom from a tiny card maker to the world's biggest games empire. Suffice to say, upon his retirement in 2002 he'd seen more than most other game execs combined.

Following the staggering success of Nintendo's original Entertainment System console (over 60 million sold worldwide), the Super NES didn't disappoint. Retaining the beige colouring but much less boxy, the SNES sported a joypad innovation in the form of shoulder buttons (copied by almost everyone else since). It launched with *Mario World*, a sprawling epic jam-packed with creativity and lifespan, and was soon joined by *Mario Kart* and *Zelda III* – two classics truly bordering on perfection. Nintendo shifted almost 50 million Super NESs, putting it way ahead of rival Sega.

Although the console was adequately powerful for sprite-based romps, Nintendo beefed up the hardware with extra chips in some cartridges, most notably the DSP for Mode-7 scaling effects in games such as *Pilotwings*, along with the Super FX for added 3D muscle in *Starwing*. Today, SNES emulation is a healthy scene with an assortment of apps doing the rounds, but two stand out: *Snes9x* and *ZSNES*. There's little to choose between the two: both apps boast excellent game compatibility, features galore and impressive speed. *Snes9x (see Emulator Options box below)* is a better choice for non-x86 PCs, but we'll concentrate on *ZSNES*. See *LXF50*'s HotPicks for a full review.

Floating kart capers courtesy of *ZSNES*' fantastic inbuilt code creator.

PLAYER INSTRUCTIONS

1 Grab a copy of *ZSNES*. Using the source archive on our coverdisc, you can compile it by hand, providing you have *SDL* and the *NASM* assembler installed. As an alternative, you could check out the binary packages – these are also on our disc. Once it's built and installed, running **zsnes** will fire up the emulator. Beltin'.

2 Get familiar with the GUI. *ZSNES* uses its own mini-windowing system and widgets, nested in the main window, and for the most part it operates like a typical GUI. Pressing the Esc key toggles between gameplay mode and the menus (the maximise button top right brings up a full-screen display).

3 Configure controls. If you've got a *SNES*-esque joypad, congratulations, but if not keyboard control isn't too painful after a while. From the Config menu, select Input #1 – there you can assign joypad buttons or keypresses to the *SNES* pad equivalents. Incidentally, www.lik-sang.com sells a spiffy little USB-to-*SNES* pad converter, for full authenticity!

4 Set up video and sound. These are accessible from the Config menu too, providing various options for rendering and resolution, along with sound sampling rate (knock it down a few notches on older machines). In general, the defaults are more than satisfactory.

5 Start playing! From the Game menu, use the file browser to find your ROMs, and load them up. (These will also be accessible from the arrow menu top left.) All being well, your faves of yesteryear should work without hassle – and you can zip through intros and slow parts with the ` key (top left on the keyboard).

6 Start hacking. Brilliantly, *ZSNES* accepts Game Genie and Action Replay cheat codes, and you can create new cheats yourself. Using the Search item under the Cheat menu, look for a value (decimal or hex) in the current game, and then retry the search later when it's changed. This is supremely useful for pinpointing where scores and so on are stored in memory – there's nothing like making Yoshi float in *Mario Kart*...

7 Explore other features. The Game menu holds a save/load state option; this takes a snapshot of the emulated console's RAM, allowing you to return to that exact point with one click. Additionally, networked play is available under the Netplay menu for interweb-based tomfoolery.

8 Explore the extra resources. Some useful websites: www.zsnes.com for *ZSNES*, www.snes9x.com for *Snes9x*, and www.emulator-zone.com, where there's stacks of info on SNES emulation, and other consoles.

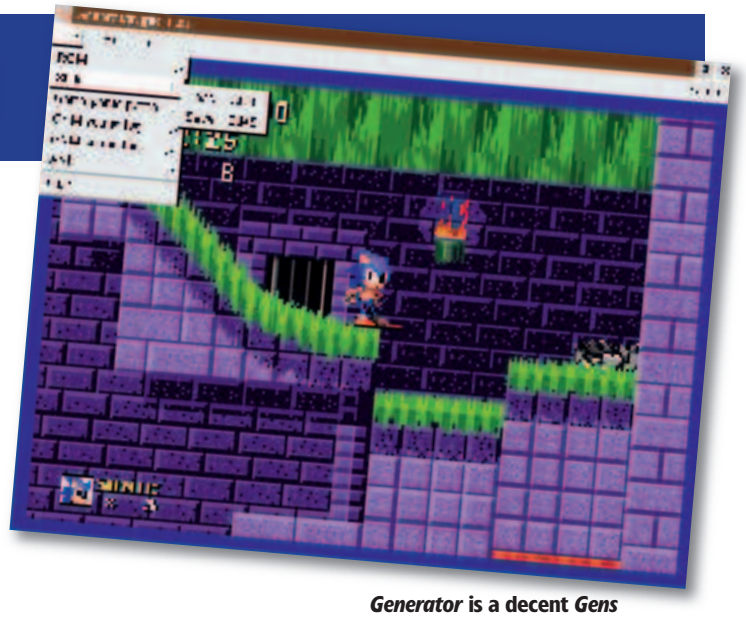
EMULATOR OPTIONS: SNES9X

Because *ZSNES*'s CPU emulation is written in assembler, it's limited to the x86 PC platform. This provides extra speed (running happily on a Pentium 150) but it's at the expense of portability. If you're on a Mac or Sun box, *Snes9x* is the better choice – it includes both assembler and C versions of the CPU emulation.

***Snes9x* is a bit harder to get started with. Options are provided at the command line, and a spot of doc-digging is required beforehand. Still, some coders have created graphical front-ends, such as *Snes9express* (www.linuxgames.com/snes9express).**

SEGA MEGA DRIVE

To be this good takes emulation...



***Generator* is a decent *Gens* alternative, although Sonic doesn't look too thrilled.**

Sega's Mega Drive (Genesis in the USA), the successor to Sega's largely successful Master System, barged on to the scene with a distinct lack of triple-A titles. A handful of arcade ports, faithful as they were, didn't inspire consumers; it was the advent of *Sonic* that really got the console revved up. Sports games from EA, together with healthy third-party support, injected more life into the sleek black box, taking sales to a respectable 30 million total units in its lifetime.

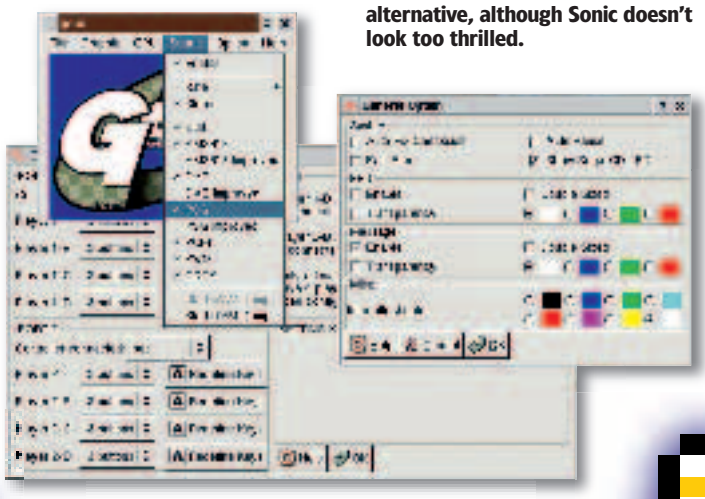
Sadly, in the later years of the Mega Drive, Sega confused the market by releasing poorly-supported addons: the Mega CD and 32X. Gamers didn't know what to opt for, and this damaged the company's image leading into the Saturn. Sega eventually left the hardware market after the (rather fine) Dreamcast suffered in sales (10 million total), but nonetheless, the Mega Drive had an excellent catalogue of games covering all bases, so emulators abound.

Our pick, *Gens*, is one of the best open source projects in this field. Combining a friendly UI with solid performance and stability, it's an ideal choice for Mega Drive emulation without the hassles of command line switches and hours of doc reading. Let's get it up and running...

SYSTEM SPEC: MEGA DRIVE

CPU: 16-bit Motorola 68000 @ 7.61 MHz
RAM: 64k, 64k video
Sound: TI 76489 and YM 2612 (10 chan)
Max res: 320 x 224 pixels
Max sprites: 80

***Road Rash* runs like a charm, perfect for those punch-other-bikers-into-cars cravings.**



***GTK 2* menus and dialogs aplenty in the *Gens* configuration, which fits well with Gnome or *Xfce*.**

PLAYER INSTRUCTIONS

1 Install Gens. We really, highly, completely, utterly, absolutely recommend avoiding the source tarball. It's so complicated you'll wish you'd coded your own emulator from scratch. On the Atari Jaguar. In Fortran. Thanks to DOS-format build files, name errors and a quadrillion other complications, getting it to compile is hair-rippingly bothersome; so if possible, go with the binaries. The Mandriva one on our coverdisc installs without hassle on Slackware (rpm2tgz), and should be coaxed into working with Debian-based distros via *Alien*. We've included other packages too.

2 Start it up. Running **gens** from the shell fires up the emulator's attractive and clean *GTK* front-end (it also depends on *SDL*), supplied with menus for configuration and a full-screen mode accessible with Alt+Enter. At any point in gameplay, hitting Esc freezes the action so you can tweak the settings or save the memory state, while pressing Shift+Backspace dumps a screenshot (BMP format) in the configuration directory – **.gens/** in your home folder.

3 Set the controls. On a typical PC keyboard, using the cursor keys for the D-pad along with A+S+D for the three buttons works well, together with Enter for Start. Handily, *Gens* emulates the six-button pad that Sega released later in the Mega Drive's life – ideal for playing *Street Fighter II* and the like. Remapping the keys is blissfully simple: just go to the Option menu and select Joypads, then click the Redefine button to tap appropriate keys at the prompts.

4 Tune sound and video. Most crucially, the Graphic menu contains a Frame Skip setting, which is essential for maintaining performance (especially on older machines). If you find games stuttering or suffering from slowdowns, try setting the Frame Skip to a higher value – it'll tell the emulator to draw the screen less frequently, thus speeding things up a few notches. Similarly, under the Sound menu you can disable some features of the audio hardware and lower the sample rate, yielding another speed boost if you're getting choppy results.

5 Nostalgia! Once it's all set up, open a game from the File menu and it'll start in the main window. *Gens* is commendably compatible; we weren't accosted by any gremlins with the games we tried – ranging from early titles to later releases with more demanding visuals – and we've used *Gens* for months without any whopping problems.

6 Dig deeper. Gens offers a bunch of extra features, such as Game Genie codes and a variety of rendering modes. More information on the app can be found at www.gens.ws, and there's a towering heap of info at www.emulator-zone.com. For an overview of other emulators, pop over to http://dextremes.com/genesis/emu.

EMULATOR OPTIONS

If *Gens* doesn't float your boat, or you're having compatibility woes with a certain game, it's worth taking a peek at some of the other emulators doing the rounds. *Generator* (www.squish.net/generator) is a promising app, but doesn't see many regular updates – there are a few patches around to give it some new features, though.

DGen/SDL, another early-stager that's lacking in updates, copes well with some popular games and can be found at http://pknet.com/~joe/dgen-sdl.html. And for those addicted to Yuzo Koshiro's marvellous music (think *Streets of Rage II*), try *XymMS* (http://sourceforge.net/projects/xymms/) – an *XMMS* plugin to play GYM files containing Mega Drive choons. It's superb.

AMIGA

A hard act to follow

The Amiga is a particularly difficult computer to emulate. Most of its features were provided by a set of custom chips, which is why it was able to give such good graphics and sounds performance with very little raw CPU power (the A500 had a 7MHz CPU).

There is really only one Amiga emulator of note: *UAE*. Others are derivations of the same code. *UAE* originally stood for *Unix Amiga Emulator*, although the U was generally considered to stand for 'Unseen' in the early days and 'Unusable' for some time after that. It was only when x86 CPUs were running clock speeds in excess of 50 times the speed of the original Amiga CPU that it became anything like usable.

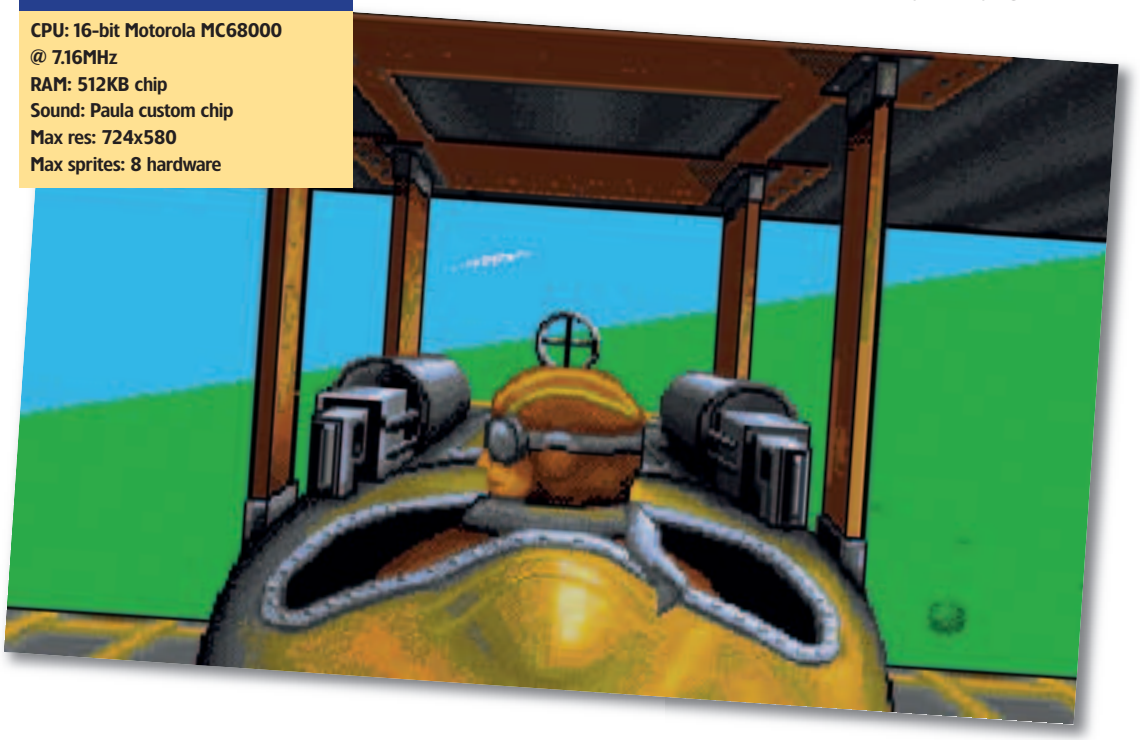
While most distros include a copy of *UAE*, it may not be the latest version. Most recent development has been on *WinUAE*, the Windows variant, and this is more up to date than the basic *UAE*. *E-UAE* brings the extra features of a more recent Windows version back to its non-Windows roots. This version is on the *LXF* coverdisc.

***UAE* has plenty of configuration options in its GUI and even more hidden in the config file. While many programs run happily with the defaults, tweaking some options can make a big difference to certain programs.**

SYSTEM SPEC: AMIGA 500

CPU: 16-bit Motorola MC68000 @ 7.16MHz
RAM: 512KB chip
Sound: Paula custom chip
Max res: 724x580
Max sprites: 8 hardware

There was honour in victory, honour in defeat, and honour in just staying alive.



PLAYER INSTRUCTIONS

1 Install *UAE* from the disc. There you'll find RPMs and an x86 static binary package as well as the source code. Installation from source follows the usual practice: open a terminal and type

```
su -
<Enter root password>
tar xjf /mnt/cdrom/Magazine/Emulate/Amiga/e-uae-0.8.27.tar.bz2
cd e-uae-0.8.27
./configure
make
make install
```

2 Set it to your liking. There are a number of options you can enable or disable during compilation. To see them, type **./configure --help**, but the defaults make a good starting point so you might prefer to stick with them.

3 Grab a ROM image. You'll need this in addition to the *UAE* program. Unlike PCs, the core of the Amiga operating system is held in a ROM chip. *UAE* includes an AmigaOS program called *transrom* that you run on a real Amiga to copy the ROM to a file, which you can then copy to your PC. It is only 512k so it can be copied by floppy disc if no other option is available.

4 Install the Amiga OS. This is required unless you only want to use *UAE* to play self-booting games (not that there is anything in the slightest bit wrong with using it like this). The OS files should be copied to a directory that *UAE* will treat as a disk drive partition. Both the ROM and the OS are copyrighted. While there are plenty of sites offering these files for download, they are not legal.

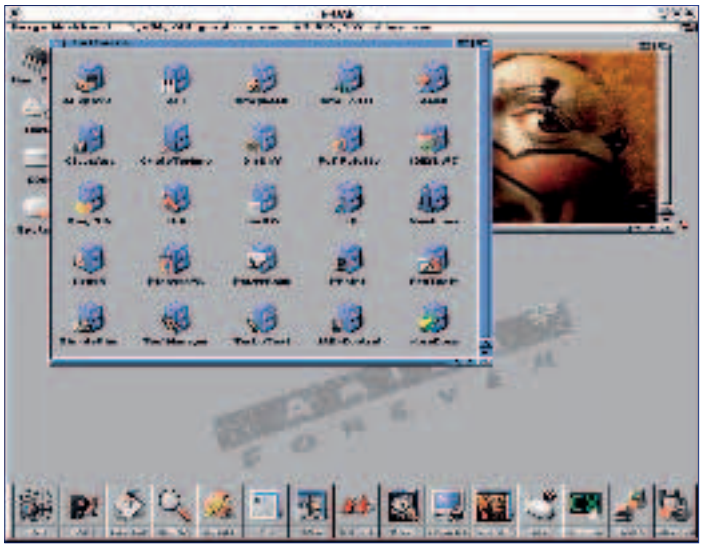
5 Start it up. Once you have *UAE* installed and the other files copied to your home directory, start the program, either by typing **e-uae** in a terminal or by adding it to your window manager's program launcher.

6 Tweaking. Some configuration is necessary before you can launch the emulator for the first time. The main requirements are to set the path to the ROM file under the memory tab and to create a virtual hard disk using the path to the directory containing the operating system files under the Hard Disks tab.

7 Set the sound. You may need to tweak some of these settings according to what you want to run on the emulated Amiga. The sound settings, in particular, can be critical. There are no hard and fast rules here, it's a matter of trial and error to find the settings that best match your system and the software you need to run.

8 Consider your CPU. The choice of CPU can be important too. For games playing, the default 68020 setting is usually best. Very old A500 programs may require a 68000 CPU, whereas more modern software often benefits from setting it to 68040. Choosing 68060 is rarely optimal, since it has a smaller instruction set than the 68040, requiring some instructions to be emulated in software. The 68060 gains much of its extra performance from a higher clock speed, a concept that's irrelevant with an emulated CPU.

9 Investigate. The configuration GUI provides access to the common settings, but more tweaking is possible if you edit the config file. See **configuration.txt** in the *UAE* source directory.



***UAE* is not limited to x86 hardware. Here it is running on PPC. This screenshot is of an *Amiga Forever* system updated to use the latest *E-UAE*.**

UAE is an interpreting emulator. That means overhead of instruction decoding is high, especially on PCs. A 65,536-entry despatch table chews up data cache, and 70 Pentium instruction dispatch slots are consumed in decoding before it starts to interpret each 68k instruction, which can do a lot more per cycle than an x86.

JUST IN TIME!

'Just In Time' (JIT) compilation of code blocks drastically reduces such overhead. Extra memory holds translated blocks and an index. After each block JIT emulators check to see if the next is pre-translated. If so, it can be run without decoding; otherwise it is translated and buffered. JIT requires at least 6MB for its tables, and prefers 12MB. Some programs benefit from even more memory.

The 64k word table is used a bit more cleverly by the JIT engine, and the Linux kernel lets the host Memory Management Unit help with the sifting. More acceleration comes from using the CMOV Pentium Pro instruction.

Despite these complications, part-translated combination code can run programs several times faster than the interpreting code. Register-bound loops may outrun the fastest 68060s on even a 1GHz PC. The JIT engine only emulates 32-bit 68k boxes and relies on defensive programming to flag the need for cache flushes, upsetting many old Amiga programs. JIT code just works on x86 hosts, though the technique may be easier on other processors, and can't handle the PowerPC code in the fastest apps written for dual-processor Amigas.

EMULATOR OPTIONS

There have been a couple of interesting commercial emulation packages based on *UAE*. *Amiga Forever* is basically *UAE* for a variety of platforms packaged on a CD complete with licensed copies of the ROM and operating system. The latest version, *Amiga Forever 6.0*, comes on a bootable CD that uses a cut-down version of Knoppix to boot Linux and then run *UAE* in full screen, giving the impression that you are booting AmigaOS on an x86 PC. In addition to the basic operating system, *Amiga Forever* comes with licensed copies of applications including *Directory Opus* and *Personal Paint*. The CD version, unlike the cheaper download edition, fills the remaining space on the disc with some video files, including the 1985 launch of the Amiga and the classic *Deathbed Vigil* video of Commodore's last days. The latter file alone would make this CD a worthwhile purchase for any Amiga aficionado. Get it from **www.amigaforever.com**.

The other commercial Amiga emulator took a somewhat different approach. *Amithlon* was created by Bernd Meyer, the man behind the JIT code in *UAE*. *Amithlon* uses a highly customised and cut-down Linux kernel to boot the emulator. The emulator was based on *UAE* but had almost all of the code to emulate the Amiga's custom chips removed, leaving just enough to provide video for the boot screen. Once the emulation was running, it acted like an Amiga with third-party video- and soundcards, using built-in drivers to access the PC's video and sound hardware. The result was an emulated Amiga that ran system-compliant software at amazing speeds, but was no good for most games. For productivity software, and more recent games compatible with third-party video hardware, it was unsurpassed and was the only Amiga emulator that did not give the impression it was an emulator.

You may have noticed the frequent use of the past tense in the previous paragraph. Unfortunately, *Amithlon* fell foul of political and commercial in-fighting with the result that it is no longer for sale. The author still takes an interest in supporting those using it, but there is currently no legal way to obtain a copy.

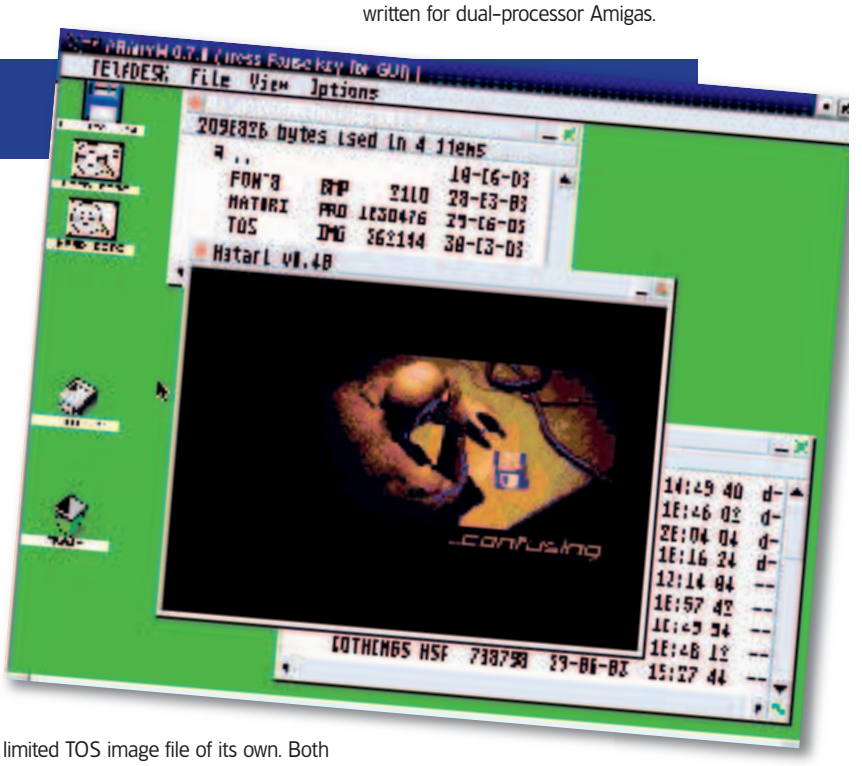
ATARI ST EMULATION

If you're a KDE or Gnome advocate, you should get some flame war fighting tips from this computing debate of 20 years ago. At around the same time that the Amiga was released, Atari released its ST computer. Both were based on the 68000 CPU, with the ST running at 8MHz while the Amiga only managed just over 7MHz, although the Amiga's custom chipset meant that it ran faster.

The main advantage of the ST, apart from it being cheaper, was that it had built-in MIDI, so it quickly became a favourite with musicians. To say that the loyalty and rivalry between owners of these two computers was intense would be a massive understatement.

Hatari is an Atari ST emulator that uses the *UAE* CPU emulation, which makes sense since both computers used the same CPU. You can download it from **http://hatari.sourceforge.net**. As with *UAE*, you also need the operating system as a ROM image. There are two alternatives here. The ideal method is to use the ROM from a real ST, using one of the utilities to rip it to a file. The other option is to use *EmuTOS*, from **http://emutos.sourceforge.net**. This is an open source replacement for the Atari TOS operating system. It is limited compared with the full TOS, but does allow many games to be played without access to a real ST ROM.

Another ST emulator is *STonX*, available from **http://stonx.sourceforge.net**. As with *Hatari*, this is available as source, to be installed with the usual **./configure && make && make install**. This comes with a limited TOS image file of its own. Both of these emulators were limited in their use without the full TOS, which is not freely available.



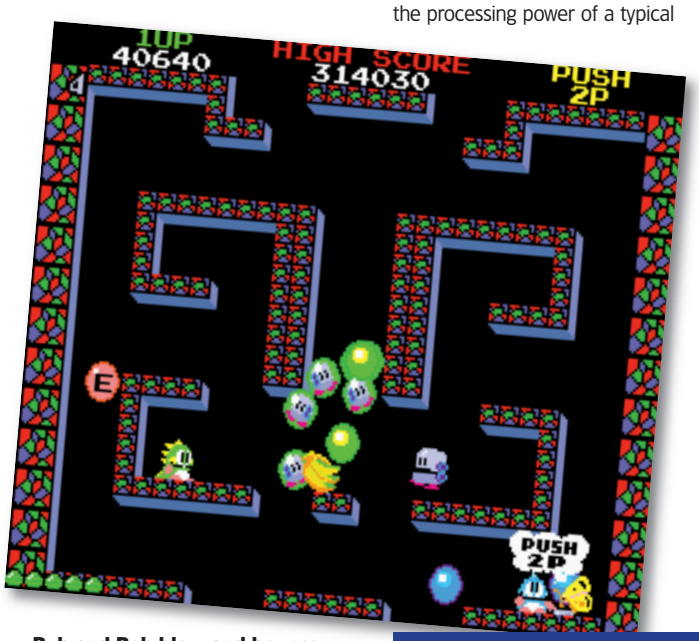
***Hatari* is one of the emulators available for the Atari ST.**

MAME

Forget the ports, play the original arcade classics.

If you want to run an emulator to play games, the king of them all is *MAME*. The acronym stands for *Multiple Arcade Machine Emulator*, which is exactly what it is. Those wooden cabinets that frequented the seedy dives in the seaside holidays of your youth housed some fairly complicated but basically similar hardware. Although not all arcade games were the same, they were close enough to allow games to be designed to work on a platform rather than having to be custom built.

The various processors used were later to find their way into home computers. Emulating this hardware is tricky, because the exact specifications did vary, but thankfully your desktop PC is likely to have about 5,000 times the processing power of a typical eighties arcade machine, so all but the latest titles work fine, thanks to the wonders of *MAME*.

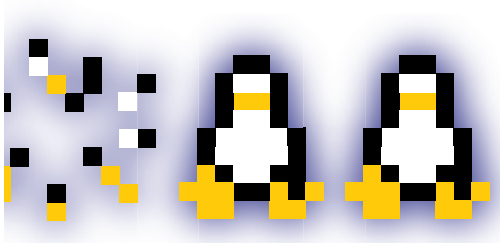


Bub and Bob blow and bounce their way to arcade fame.

Collect the powerups and turn into a wolf. It made sense at the time.

Our favoured version of *MAME* at the moment is *AdvanceMAME*. It is fairly up to date with the very latest *MAME* code releases, but it is easy to compile, works across a number of platforms and has a nice front-end component. You'll find the *AdvanceMAME* and *AdvanceMENU* front-end code on the discs this issue..

Press Tab at any time to get the main *MAME* menu. If you can tear yourself away from *Mr Do*.



MAME KEYS

You won't get too far with *MAME* if you don't know which keys to press! As well as various systems options, keys are still the best way to play some of these arcade classics.

5.....Insert coin
1.....Start 1 Player game
2.....Start 2 Player game
Arrow keys.....Player 1 direction
Left Control.....Player 1 Button 1
Left-Alt.....Player 1 Button 2
P.....Pause
Tab.....*MAME* menus

PLAYER INSTRUCTIONS

1 Get hold of the *MAME* source code. There may be packages available for your distro, but we have chosen to show you how to compile from source. In this case it is fairly straightforward. The only major dependency is ***SDL***, which is probably installed on your system anyway, certainly if you play any games.

2 Unpack *AdvanceMAME*. Log in as root and enter the usual:

```
tar xvz f advancemame-0.96.0.tar.gz
cd advancemame-0.96.0
```

3 Select a location. By default, the game will install itself in **/usr/local/bin/** and the data directories will be in **/usr/local/share/advance**. If you don't want to use these defaults, you need to pass some options to the configure command. Enter

```
./configure --help
```

to see the options. Otherwise you can just type

```
./configure && make && make install
```

All that could take a while. *MAME* takes around ten minutes to build on a 2GHz processor. Once it has finished, *MAME* itself will be installed and ready to go. However, if you want to run your emulation the easy way, you'll want to check out *AdvanceMENU* too *(see step 5)*.

4 Add some games. Before you can really use *MAME*, you need to add some recognised ROM files to the directory. As a start, you could try to find *Robby Roto*. The owner of this game has put it into the public domain, so you can legitimately use it even if you don't have the original ROMs. You need to copy the ROM file to **/usr/local/share/advance/rom/** for it to be picked up by *MAME*. From the command line, you can then run *AdvanceMAME* directly, just by giving it the name of the ROM file, for example:

```
advmame  robby
```

5 Install *Advance MENU*. *MAME* is all very well, but it's a little tough to remember all those ROM names. The *AdvanceMENU* software comes to the rescue here. It can automatically check for installed ROMs, and shows a friendly user interface to help you choose the game you want to play. You can even take screenshots from the games (F12) to appear as thumbnails in the listings. Here's the quick way to set up *AdvanceMENU*. As before, get hold of the latest source code, switch to the root user, then unpack it with

```
tar xvz f advancemenu-2.4.9.tar.gz
cd advancemame-2.4.9
```

As usual, if you want to change any of the install parameters, you should check out the config options. Otherwise just run

```
./configure && make && make install
```

You should now be able to run *AdvanceMENU* from the command line, like so:

```
advmenu
```

The first time you run the software it will create the default config file. Run it a second time to use the software. Hit the backtick (`) key for the menu and options.

BLOCKBUSTING GRAPHICS

The gameplay of many arcade classics can hardly be improved, but the graphics are generally blocky and awful. Well, that's the way they look on your high-resolution monitor anyway. The original games derived a lot of benefit from being displayed on TV screens, which makes the edges softer. In fact, a lot of the graphics were designed with that specifically in mind. On your pin-sharp LCD display, the blocky graphics don't look quite so good, especially when you magnify the display so it takes up more than a postage stamp-sized slice of your screen.

MAME now includes a variety of effects to make the blocky nature of the graphics less noticeable. There are a variety of tricks to choose from, including ones that simply blur the graphics and those that attempt to interpolate the pixels, with mixed results.

The effects tend to work best when specified on a game-by-game basis. A game like *Mr Do's Castle*, on balance, probably does look better with some sort of effect, as we show here. But try applying smoothing to something like the classic *Space Invaders* game, and it just doesn't look right.

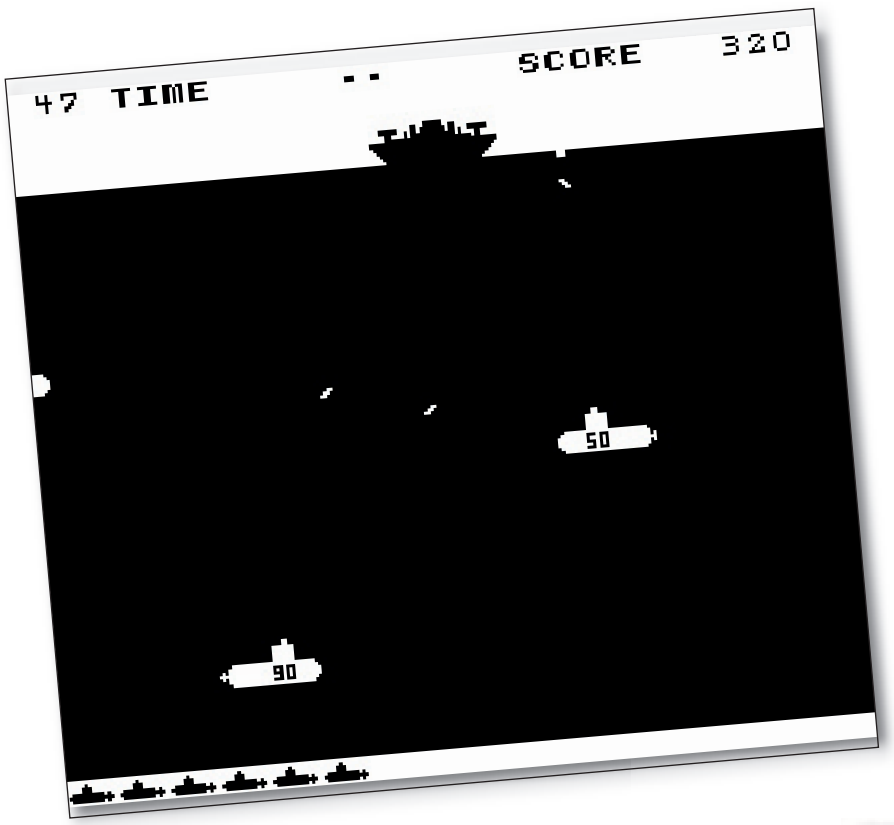
If you want to play with the effects yourself, hit Tab in the running *MAME* game and use the cursor keys and Enter to select Video from the menu. About halfway down you'll find the Resize Effect entry – just use the left and right cursor controls to change the current setting. To see the kind of effects possible, here are four renditions of a *Mr Do* character in action.

As nature intended, but the blocky nature becomes very visible when magnified.

One of the filter effects, this blurs the image to appear more like a TV display.

Interpolated pixels mean smoother edges, but it doesn't always work this well.

The Auto option will try to pick the best effects.



It may be simple, but *Depth Charge* guzzled 10p pieces by the bucketload.

Apart from the excellent music, this is how we remember *Outrun* best of all... Upside down.

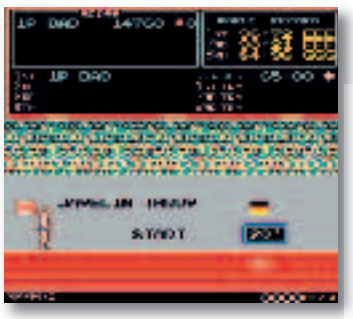
LEGALITIES

As with the other emulators we feature here, you should only run games on *MAME* that you have permission or the rights to use.

In the case of arcade machines, this usually means possession of the ROMs themselves, although some companies have released older ROM files that are effectively in the public domain. However, we cannot be held responsible for your actions – the law varies considerably around the world, so please make sure you play within the laws in your part of the world.

Some ways of legally obtaining ROMS:

- Buy the original hardware. This is often for sale on websites like eBay, and some boards with ROMS in can be had for a few pounds.
- Buy a compilation CD. Many authorised 'arcade classics' games, like those for the PS2, and some PC games are actually versions of *MAME* or similar emulators and include the ROM files.
- Obtain permission from the licence holder. Some old ROM files, such as *Robby Roto*, have been released into the public domain.



Your fingers may never recover if you subject them to *Track and Field* again. Is that West German female athlete looking a little bulky to you?

THE GREATEST GAMES EVER?

Is it *SWOS* on the Amiga, *Mario Kart* on the SNES, or *Manic Miner* even? Wars have (probably) been fought over the greatest ever games, but we're keen to find out which ones you reckon are the best. Email us at lxf.letters@futurenet.co.uk or pop in to the forums at www.linuxformat.co.uk to let us know.

Your favourite? We won't judge.

There's more to emulation than just games...

APPLE MAC

Power- or 68k-based, new or old, it can be emulated.

***SheepShaver* can use real Mac OS partitions.**

Although emulators that ape 'Classic' Mac hardware and virtualisations that let you run Mac OS on Linux on PPC hardware have been available for some time, it is only recently that Linux users have been able to run PPC flavours of Mac OS on non-PPC hardware. Thanks to recent development of the existing *SheepShaver* emulator and the new offering *PearPC*, this is now possible, although maybe not yet practical.

Emulating the PowerPC processor is a tough job, and requires some beefy hardware. While a fast x86 system can cope well with running Mac OS 9 and earlier versions, the more resource-hungry OS X is going to place a strain on even state-of-the-art PC hardware. Which rather begs the question, why bother? After all, a second-hand G3 Mac will run OS X better and more cheaply. The answer: because you can.

Option 1: SheepShaver

www.students.uni-mainz.de/bauec002/SheepShaver.html

SheepShaver started life as a commercial PowerMac virtualisation for BeOS on PPC hardware. The odd name is a pun on *ShapeShifter*, an earlier program from co-author Christian Bauer which allowed you to run a 68k Mac environment on Amiga OS. *SheepShaver* appeared on Linux/PPC in March 2001, and became an open source project in 2002, integrating code from Bauer's portable 68k Mac emulator, *Basilisk II*.

Since then a PPC emulator has been added, allowing *SheepShaver* to run on any processor architecture, albeit with greatly reduced performance. A JIT compiler is included for x86 systems only.

SheepShaver takes a higher-level approach to emulation than *Mac-On-Linux* or *PearPC*, the other emulators we look at here. Instead of emulating the Mac system hardware, it patches the Mac OS Toolbox ROM to create an environment where it can run on the host operating system. This has some disadvantages – you must have a Mac OS ROM to use *SheepShaver*, and you cannot run anything other than Mac OS 8.x or 9.x on it – but there are advantages too. For example, with *SheepShaver* you don't have to install any Mac OS drivers to use bridged hardware devices because this is all handled when the ROM is patched at startup. It also allows some neat feature such as clipboard sharing and being able to mount a host folder under Mac OS to share files.

You can use real Mac OS partitions and image files with *SheepShaver*, and networking is supported either via a virtual tunnel to the host using the Linux TUN/TAP driver or a dedicated host Ethernet card directly (using a special kernel driver).

We tried the latest x86 binary snapshot and compiled our own binaries from *CVS* for x86 and PPC systems. Unfortunately, the PPC build wasn't stable enough to test, and there are no pre-built PPC binaries available. The x86 builds were more robust, but were prone to freezing up on occasion.

The first hurdle in running *SheepShaver* is that it's very fussy about ROM images. You can use an Old World ROM, but peeking at the source code revealed that Old World ROMs from TNT, Alchemy, Zanzibar, Gazelle and Gossamer motherboards are supported. We tried grabbing images from a PowerMac 5400 (Alchemy) and a 6500 (Gazelle), but the ROM dumper just produced garbage. It's equally pernickety about New World ROMs, barfing on images on both the OS 9.0 and 9.2 install discs. We finally used a New World ROM image downloaded from an Apple update, and that allowed to boot and install Mac OS 9.0.

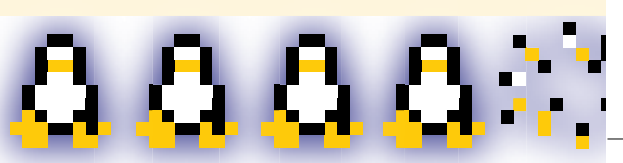
On our test machine, an Athlon XP 2200+, *SheepShaver*'s performance was impressive. Although we ran no proper benchmark, in use the system felt rather like a 200MHz PPC box, albeit with much better I/O speed than

NEW VERSUS OLD

Until Steve Jobs unveiled the iMac, all Macs had a substantial portion of their operating system in ROM on the motherboard. From the iMac onwards, the Mac firmware (known as Open Firmware – think x86 BIOS, but much, much better) is able to boot an image of this ROM from disk or CD-ROM, or indeed any other operating system such as Mac OS X or Linux. To mark the change in how they boot, these new Macs are termed New World machines, and the beige box Macs that came before became known as Old World.

Why should you care? Well, Mac OS 8.5 to 9.2 included a copy of this ROM image on the install disc. This means you no longer needed to extract a ROM image from a real Mac to run Mac OS on a Mac emulator. In fact, you don't need a real Mac at all, just the OS, which can be bought separately (although be aware that it is against the terms of the End User Licence Agreement to install Mac OS on non-licensed hardware).

If you do need a ROM image from an Old World Mac you can download the *ROM Grabber* tool from the *Mac-on-Linux* web page to dump out your system's ROM a file. You need a version of Mac OS older than 8.6 to do this, and you must disable extensions while booting (by holding down the Shift key). Despite complying with these directions, we were still unable to successfully grab a ROM image on any machine we tested.



such a machine would have. If the stability problems can be ironed out, then *SheepShaver* provides a usable environment for running OS 9 applications on x86 hardware.

Option 2: Mac-On-Linux

www.maconlinux.org

Mac-On-Linux is an open source PowerMac virtualiser that's been around since 1997. Releases from lead developer and maintainer Samuel Rydh are sporadic, and the latest version at the time of writing is over a year old. Despite this, *MOL* provides a feature-rich and flexible environment for running other PowerPC operating systems on a Linux/PPC host.

User-mode software runs at full native speed under *MOL*. Only protected opcodes such as MMU instructions are emulated, which *MOL* accelerates via a special kernel module (which also accelerates access to the emulated framebuffer).

Unlike *SheepShaver*, *MOL* emulates the rest of the Mac hardware, too. *MOL*'s booter provides a subset of the functionality of the OpenFirmware boot PROM found on a real Mac, and allows *MOL* to boot a New World Mac OS ROM directly from CD-ROM or from an installed Mac OS partition. *MOL* can also boot OS X, ELF images such as a Linux kernel, and ROM images extracted from Old World machines. The boot method is passed as a startup parameter, and each boot method and the environment it creates can be configured separately.

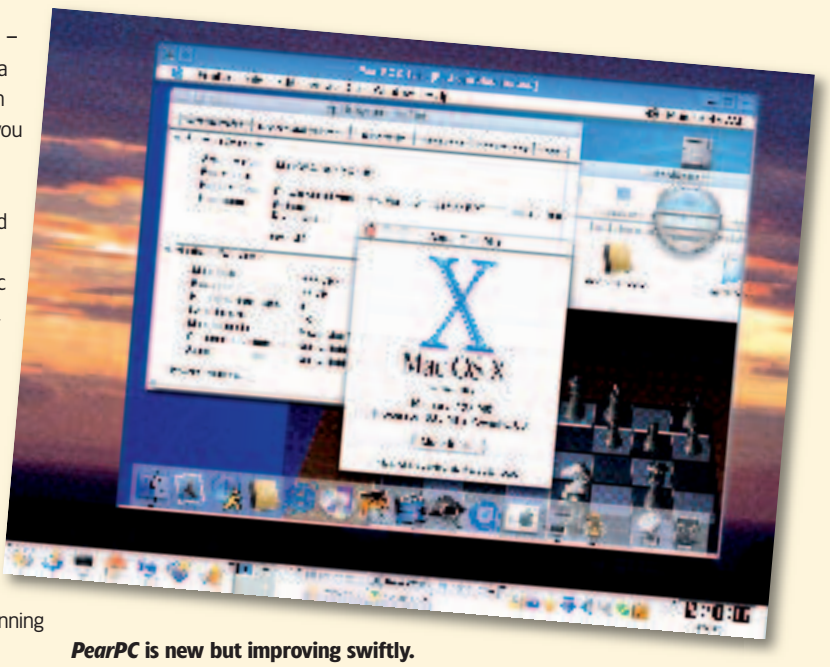
MOL doesn't employ hard files – an Amiga filesystem contained in a single file. Instead, you tell it which host block devices to use. Either you specify the partitions to mount individually or give *MOL* an entire disk and it will use any unmounted partitions on it. This is handy for users who dual boot between Mac OS or OS X and Linux, since *MOL* can boot a native install of Mac OS. *MOL* also supports SCSI and USB emulations. Network access is provided by the *SheepShaver*'s **sheep_net** kernel module or via a TUN/TAP tunnel. Video output can be to *X* or the Linux framebuffer, or the display can be exported via a built-in VNC server and accessed remotely. Running in the framebuffer is fastest.

Once installed the package is easy and direct to use. It provides a virtual environment for running Mac OS and OS X that performs equal to and in some cases better than native speed. The only thing that is noticeably slower is video. Linux I/O drivers tend to perform better than their Mac OS counterparts, so disk and network access tend to be faster in *MOL* than on the same hardware running Mac OS or OS X.

Option 3: PearPC

http://pearpc.sourceforge.net

New to the PowerPC emulation game, Sebastian Biallas's *PearPC* is attracting a lot of attention. It's a PowerPC system emulation with a fast JIT compiler for x86 systems and the first emulator that is capable of running Mac OS X on Windows and Linux.



***PearPC* is new but improving swiftly.**

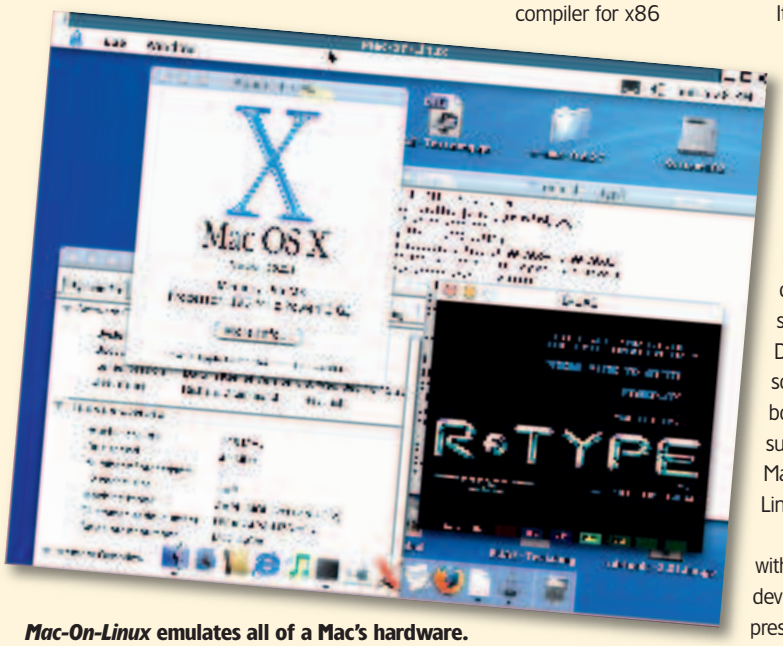
The pace of development has been swift, and *PearPC* is becoming more robust, faster and more useful.

It is, though, still a little light on features. It can only use hard files for disk emulation, but it can use a real CD-ROM drive or mount an ISO image as an emulated drive. Hard disk emulation still has problems, though, and you need to partition your hard disk by booting the Darwin/PPC distribution and using the shell-based *pdisk* tool. The OS X Disk Utility doesn't work fully for some reason. Mac OS 9 cannot be booted yet, and we've had little success when trying *PearPC* on Mandriva Linux, Ubuntu or Debian Linux either.

Network emulation is supported with the ever-popular TUN/TAP device, and interestingly *PearPC* presents to the emulated environment as an RTL8139 or 3com 3c905 chipset. Thus Mac OS X can detect it automatically, without requiring you to install a driver. Display output can be via *X* or *SDL*.

The speed of *PearPC* impresses, but it's not quite fast enough yet. We tested it on an Athlon XP 2500+ and on this machine it teetered on the edge of usability. Installing Mac OS X 10.2, leaving out non-essential items such as printer drivers and locales, took over an hour and a half from start to finish, which actually isn't too shabby, compared with the speed of a slow Mac.

The installed OS was quick to boot but in actual use, OS X was slower on this setup than on any real Mac we've tried. For instance, the system had a hard-time keeping up while you typed, and the operation of GUI controls was sluggish enough to be annoying. Since our test system was hardly cutting-edge, expect *PearPC* to prove a lot more comfortable on more modern hardware. LXF



***Mac-On-Linux* emulates all of a Mac's hardware.**

EMULATING A 68K MAC

Several emulators for pre-PowerPC Macs are available on Linux. *Basilisk II* (see www.students.uni-mainz.de/bauec002/B2Main.html) can emulate a range of Mac hardware, running code natively on a 68k host or by using an emulator based on *UAE*'s 68k emulator. A JIT compiler is included for x86 and AMD64 hosts, which can run Mac software much faster than the fastest 68040 Mac did. *SheepShaver* shares code with the *Basilisk* project, and they sport similar features.

VMac **(www.vmac.org) emulates only Macs with a 68000 processor, such as the original Mac, SE and Mac II. *Mini Mac* (http://minivmac.sourceforge.net) started as a pared-down *vMac*, but since the parent project hasn't seen much development recently, *Mini vMac* is actually more useful.**

Remember that you need a ROM image from a real Mac for these emulators to work, and since few 68k Macs are in service now, these may prove hard to come by.