FOSSpicks

Sparkling gems and new releases from the world of Free and Open Source Software

Mike Saunders has spent a decade mining the internet for open source treasures. Here's the result of his latest haul…

Connection problem solver

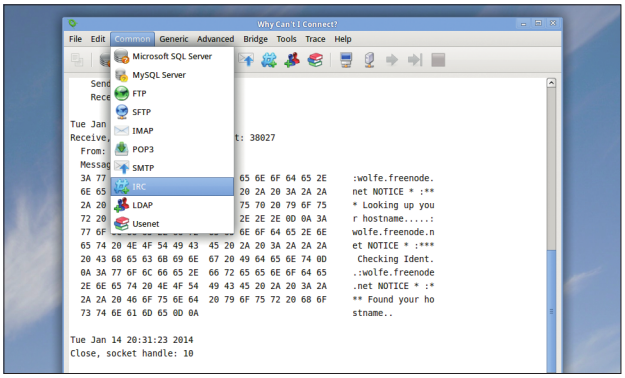
WHY CAN'T I CONNECT 1.6.2

http://wciconnect.sourceforge.net

Gimp, Firefox, Evolution, Scribus – all great free software projects, but their names say very little about what they actually do. That's not a criticism we can level at Why Can't I Connect (WCIC), however, because it does exactly what it says on the tin. WCIC attempts to help you pinpoint problems when you're connecting to servers across the internet, providing more information than you might otherwise receive from your usual applications.

Say for instance that you're trying to set up a mail program, and you can't connect to an IMAP server. Your mail client might just throw up a generic "Connection failed" error message, which is about as much use as a chocolate teapot. Where exactly is the connection failing? Is the problem with the IMAP server, or something else in between on the net? If you're massively knowledgeable about the particular protocol you're using, you might be able to Telnet into the remote machine and work it out from there, but not everyone can do this.

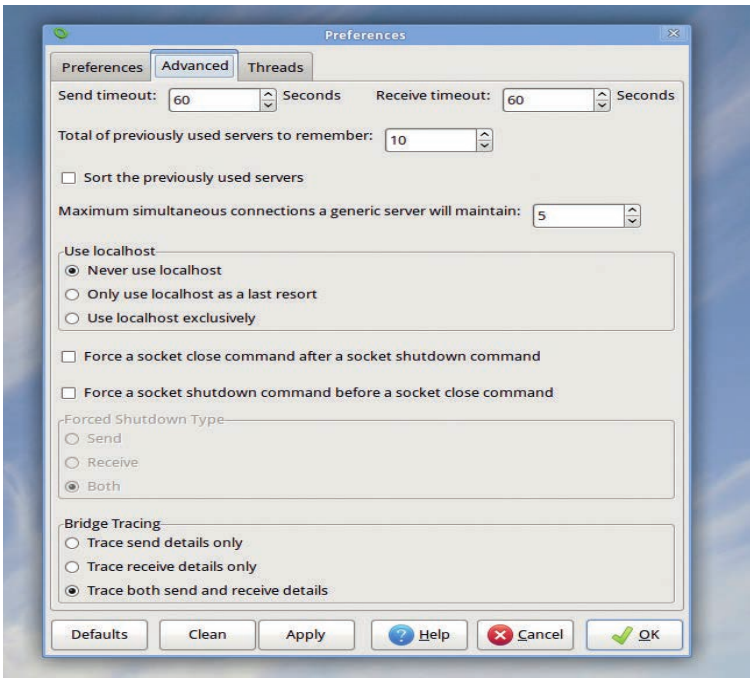
WCIC makes things much easier. You tell it to initiate a connection with a remote machine, and it steps through the process, spitting out lots of feedback and detailed descriptions of error messages. It supports a bunch of protocols out of the box: MS SQL, MySQL, (S)FTP, SMTP, IMAP, POP3, IRC, NNTP and LDAP. Just supply a hostname/IP address and a port number, and WCIC will sort out the rest.



Never struggle with lame error messages again: WCIC tells you what's really happening with your connections.

If you want to investigate connection problems with a different protocol, you can go into the Advanced menu and look through the connection process step-by-step, sending textual data from a file to see how the remote server responds. It's even possible to set up WCIC as a bridge between a client and a server, so you can look at the data that the machines are exchanging in real time.

WCIC is one of those helpful tools that we'd love to see included by default in mainstream desktop distributions - so make it happen, distro devs!



Set timeout limits and fine-tune how WCIC interacts with remote servers via the Preferences pane.

Free-form data organiser

TREESHEETS 26-10-2013

http://strlen.com/treesheets

Take a spreadsheet, mix in some mind-mapping elements, add a few to-do list and text-editor features, and what do you get? The result might sound like a hideous mish-mash of programs, but it's actually a rather awesome fusion that works really well. TreeSheets' author describes it as "suitable for any kind of data organisation", such as calendars, to-do lists, project management charts and brainstorming diagrams.

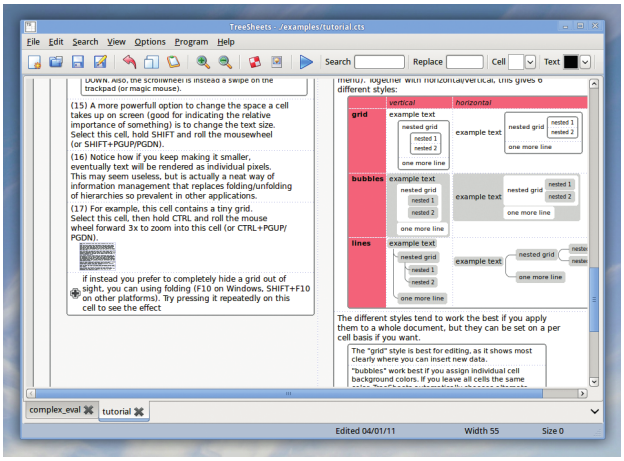
The key to all this flexibility is hierarchy: items can be embedded inside other items.To get it running you just need to grab the **treesheets_linux.tar.gz** file and extract it – this produces a directory called **TS**. In here, run **./treesheets**, a pre-compiled binary. This worked without any fuss on our Xubuntu 13.10 test machine, and providing you have GTK 2 and its dependencies installed, you shouldn't have any problems. (If the binary doesn't work, you can still compile from source.)

Excellently, TreeSheets is supplied with an in-depth tutorial that doesn't just show you how to do things, but encourages you to experiment by hand. It's shown automatically on the first launch; to bring it up at another time, hit F1. Essentially, TreeSheets works as a souped-up spreadsheet, letting you enter data into cells and move them around. Click on the edge of a cell and its border turns into a dotted line: start typing and new data is added into a new cell. Hit the Insert key (or go to Edit > Insert New Grid) to add data to a new table – like a spreadsheet inside a spreadsheet.

Lookin' good

You can format these grids in various ways via Edit > Layout Render Style. With these features alone you can come up with novel ways to present data such as contacts lists, to-do lists, project plans and even algorithms – see the **examples/** folder for inspiration.

But TreeSheets can do a lot more. It's possible to filter content based on edits, eg showing only the cells that have been edited in this session. The early phases of a simple programming system have been included, where cells can be marked as operations (eg addition or multiplication) and then generate results from neighbouring cells. Once you have everything in place, click the Run button on the toolbar to calculate results. The **complex_eval.cts** example sheet demonstrates these features well.

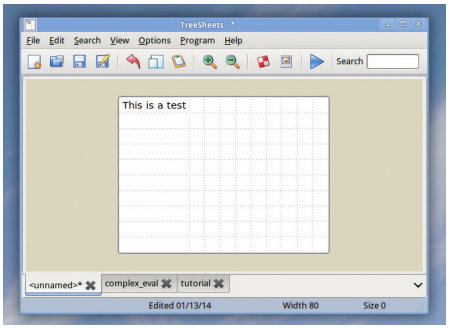


The built-in tutorial is one of the best we've ever seen, getting you hands-on with the program right from the first step.

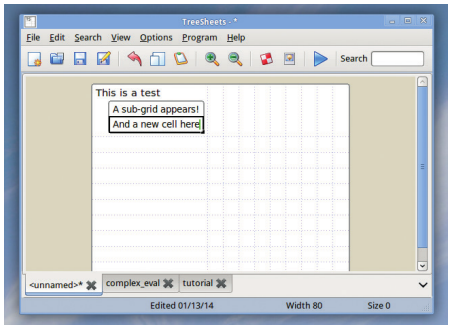
> **"The key to TreeSheets' flexibility is hierarchy: items can be embedded inside other items"**

Ultimately, TreeSheets is only limited by your imagination. It might seem like an app without a specific purpose at first, but once you've browsed the examples you'll see how the program's features could be applied to your own data. We've already started using it as a to-do list manager on steroids, and as the programming features get more fleshed-out, its range of uses will be nigh-on endless.

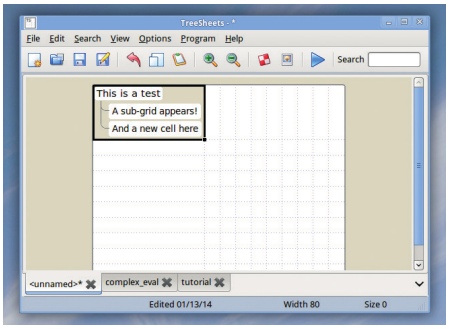
How it works: Adding data



1 Create a grid
Click on File > New to create a blank 10x10 cell grid. Start typing, and the first cell will expand to fit the data that you enter.



2 Add a sub-grid
Click Edit > Insert New Grid to create a grid inside the current one. Type some text, click on the lower edge, and type again to add a new cell.



3 Format it
Click on blank space in the outer grid, then Edit > Layout Render Style > Vertical Layout With Line Style to display the items in a tree-like format.

Graphical disassembler

EMILPRO 3.0

www.emilpro.com

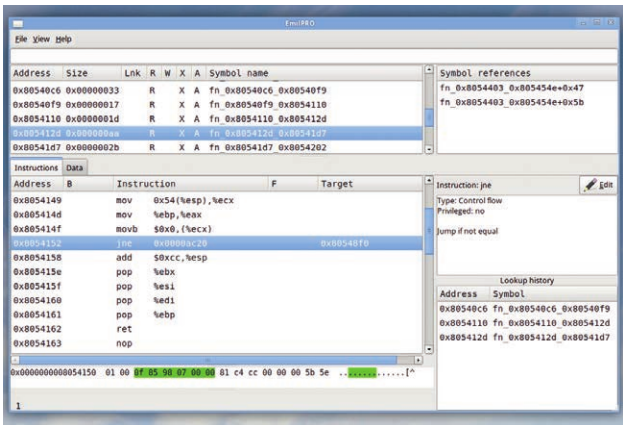
Some under-the-hood things are worth understanding, even if they're not especially useful on a day-to-day basis. One of these, if you're a programmer, is assembly language. Sure, very few people write substantial code in assembly nowadays (apart from embedded device and driver developers), but it's worth understanding what goes on after you compile your high-level code.

EmilPRO is a GUI disassembler: it takes binary machine code executables, identifies the chunks that are code (instead of data), and shows the human-readable CPU instructions contained therein.

The README file explains the dependencies required and how to compile the program. Once you have EmilPRO up and running, go to File > Open and locate a binary executable on your filesystem. On a Linux box this executable will be in the ELF format, but EmilPRO can also read Windows and Mac OS X executables using its own version of Binutils, which it builds during the compilation process.

The interface

The top-left panel shows a list of sections and symbols in the executable, while the panel to the right refers to the currently selected symbol. Underneath, on the left, you have the disassembled code (or a hex dump if it's a data section), and the right-hand side shows some information about the currently selected instruction. We're not sure how useful this is – after all, if you don't know what the instructions



Looking inside /bin/ls: the code here shows a subroutine, which hands control back to the calling code with the "ret" (return) instruction.

mean, you're probably not interested in looking at assembly language – but from inside the app you can submit more relevant descriptions to EmilPRO's website.

EmilPRO works well enough, although the interface could do with some polish. The program's author admits that he's not an expert with GTK, so if you are, dear LV reader, hop over to **https://github.com/SimonKagstrom/emilpro/wiki/TODO** to lend a helping hand.

Super-simple cron job creator

EVERY 0.1.0

https://github.com/iarna/App-Every

One of the best things about Linux (and most Unix-like systems in general) is the widespread use of plain text configuration files. Look in **/etc**, for instance, and you'll see that almost everything is readable and editable in a standard text editor. Contrast this to Windows, with its dreaded registry... Ugh. Let's not go there.

It's not a total utopia in Linux though: most configuration files are text-based, but the syntax and formatting in them can vary wildly. If you're a seasoned Linux admin then you can probably write cron entries in your sleep – but if you always end up having to consult the manual page, Every is for you. It's a small Perl script that lets you enter human-readable descriptions of tasks that should be executed at specified intervals, and it updates your crontab accordingly.

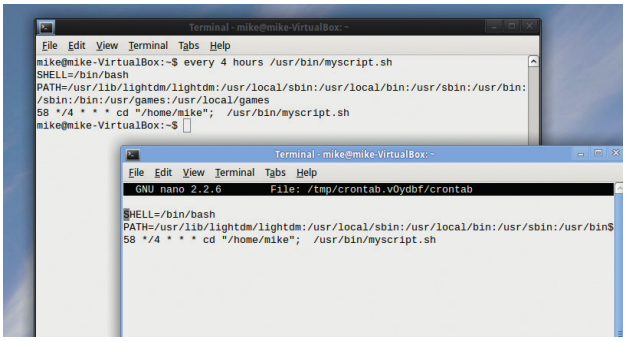
Get the latest release like so:

```
wget 'https://raw.github.com/iarna/App-Every/master/packed/every'
```

Then make it executable (**chmod +x every**) and copy it into **/usr/bin** or somewhere else in your **$PATH**. Using it is really simple: let's say that you want to run **/usr/bin/myscript.sh** every hour. All you need to enter is this:

```
every hour /usr/bin/myscript.sh
```

And that's it – you don't get much more human readable than that,



Every shows you its generated entry before adding it to the crontab – use the --dry-run option to preview the entry.

right? You can replace "hour" with "minute", "day", "week" or "month", and even specify a day of the week (eg "every thursday"). Additionally, you can specify units like so:

```
every 23 minutes /usr/bin/myscript.sh
```

When you run Every, you'll see that it spits out a crontab entry, and it also automatically updates the crontab for your user account (edit it manually with **crontab -e**). Every sets up the $PATH and your filesystem location exactly, so you can guarantee that the command will run exactly as intended.

> **"Most config files are text-based, but their syntax can vary wildly"**

Log file monitor

BEOBACHTER 1.7.8

http://sourceforge.net/projects/beobachter

Keeping tabs on log files is just one of the many jobs a hard-working sysadmin has to do. The trusty **tail** command line tool (with the **-f** flag) does a decent job here, but it's plain, and doesn't help you to spot critical information when text is whizzing by. A more elegant solution is to use a log file monitor, and Beobachter (from the German "*beobachten*, to observe) has a few aces up its sleeve.

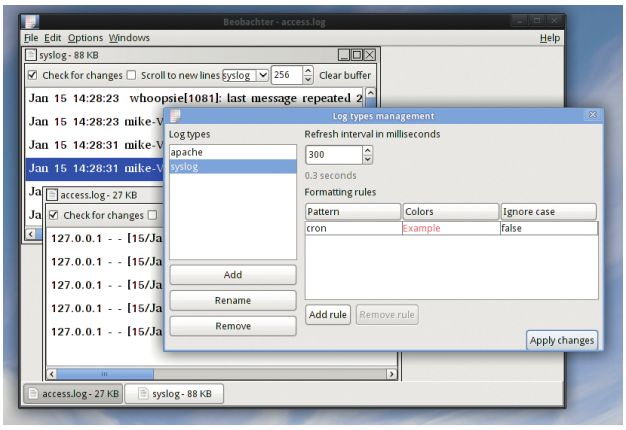
To run Beobachter, grab the **.jar** file and enter:

```
java -jar Beobachter-1.7.8-jar-with-dependencies.jar
```

Click File > Open and select a log file (eg **/var/log/syslog**). Beobachter will then show updates to the log file in real time, so as soon as anything is added to the log, you'll see it in the window.

Beobachter can view multiple log files simultaneously; you can save your session (so that all your log files are re-opened when you next start the app) via the File menu.

Where Beobachter really comes to life is with the formatting rules. Under Options > Manage Log Types you can create custom rules to apply to specific log files (or groups of log files), so that lines are highlighted when they contain specified words or regular expressions. For instance, you might be watching a log that's updated often, so you don't have time to read every line, but you want to be alerted when the word "error" crops up.



Its default interface is very ugly (it's written in Java), but you can tell it to adopt some elements of your GTK theme in Preferences.

> **"Where Beobachter really comes to life is with the formatting rules"**

With the formatting rules feature, you could set lines that contain the word "error" to have a red background, so they're easier to spot. After some tweaking, you can have a full-screen Beobachter session set up with multiple log files being viewed and errors leaping out of the screen. Not only does this make your life as an admin easier, it looks pretty l33t too.

Browse inside zip files

FUSE-ZIP 0.4.0

http://code.google.com/p/fuse-zip

Most modern file managers let you browse inside compressed archives. Gnome and KDE have services to give seamless access to compressed archives, for instance, but standalone non-Gnome/KDE programs can't use them without pulling in loads of dependencies.

fuse-zip fixes this problem by letting you manually mount **.zip** archives as normal directories, without the need for any major dependencies. The only thing it requires is FUSE (Filesystem in Userspace) which is included by default in many major distributions. To build fuse-zip, you'll need at least version 2.7 of **libfuse-dev** and version 0.11 of **libzip-dev**. On our Xubuntu 13.10 test box, entering **make release** to build it resulted in an error that UINT16_MAX was not defined; we fixed it by adding this to the top of **lib/fileNode.cpp**:

```
#define UINT16_MAX 65535
```

Once you have it built, run **make install** as root and you're ready to go. Mounting a zip file in a directory is as easy as pie:

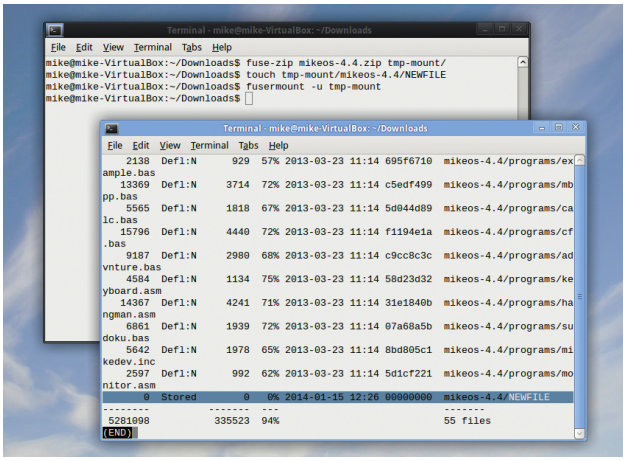
```
mkdir mytmp
fuse-zip somefile.zip mytmp
```

Now you'll be able to seamlessly browse the contents of **somefile.zip** via the **mytmp** directory. Any edits that you make to files will be stored back in the **.zip** file when you unmount the directory, which you can do like this:

```
fusermount -u mytmp
```

fuse-zip's developer claims that it's faster than the Gnome and KDE equivalents, with plenty of stats to back it up. You can even use it to create new zip files, eg **fuse-zip newfile.zip some-directory**. When you unmount **some-directory**, **newfile.zip** will be created.

It all runs like clockwork, so if you've been avoiding the heavyweight desktop environments but you still like this feature of their file managers, give fuse-zip a go.



In the background terminal we mount mikeos-4.4.zip, add NEWFILE to it, and unmount it. And the foreground terminal shows it has been added to the .zip. Awesome.

Crazily tiny PC emulator

CABLE3

http://ioccc.org/2013/cable3

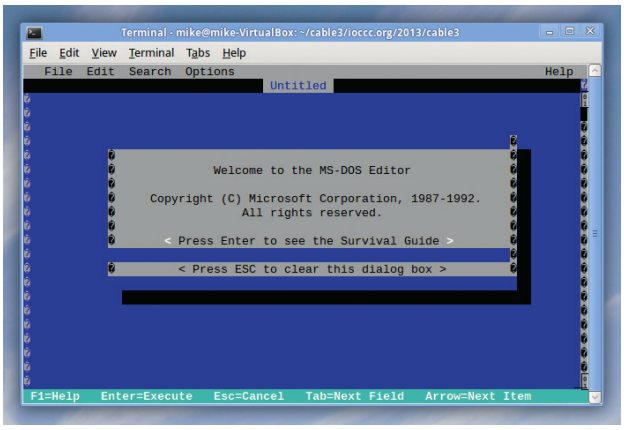
Look at this snippet of source code and see if you can guess what language it is

```
*i+=262*o*z(F((*E&15)>9|42[E]),*E&=15))
i(SP,(w(7),R&&--
1[i]&&o?R++,Q&&Q++,M--:0))DX()
{$,O*=27840;O--;
```

If you're feeling particularly snarky, you might say "It's Perl, with some extra formatting to make it more readable than a typical Perl program". But no, it's actually C - and completely valid C. Each year, the International Obfuscated C Code Contest (**www.ioccc.org**) asks programmers to submit the most weird, messed-up and preprocessor-abusing code possible, and some of the results are truly astounding.

The above snippet is from Cable3, a fully fledged PC emulator that's generated from a single 4043-byte C source code file. It's the obfuscation that keeps the code so small – in human-readable C it'd be a lot bigger – but regardless of how the code gets mangled, it's an impressive project nonetheless. All you need to build it is the SDL development libraries, so once you have them installed, just enter **make**. You can then run the emulator by supplying a BIOS image and a floppy disk image at the command line; see the **runme** script for an example.

Cable3 emulates a mid-1980s-era PC, complete with an 80186 CPU, 1MB of RAM, a Hercules graphics card, a floppy drive and a hard drive controller. So it's not particularly powerful compared to the likes of QEMU and VirtualBox, but it's capable of running lots of programs such as early versions of Lotus 1-2-3 and Flight Simulator. It's even possible to coerce Windows 3.0 into running – don't expect great performance though.



It's a bit rough around the edges, but squeezing an entire PC emulator into 4k of code is an astonishing achievement.

> **"Cable3 emulates a mid-1980s-era PC, complete with an 80186 CPU"**

MS Paint-esque graphics editor

PAINT.JAVA 0.9C

https://github.com/HeroesGrave/Paint.JAVA

Microsoft Paint is a pretty rubbish image editor, but if you've ever spent time in Windows, you've probably used it in an emergency. It's basic, it's uncomplicated and it has hardly any features – but it's always there. An enhanced version called Paint.NET has been in development over the past few years, with a similar interface but much more functionality included, and now we have a Java-based clone of it.

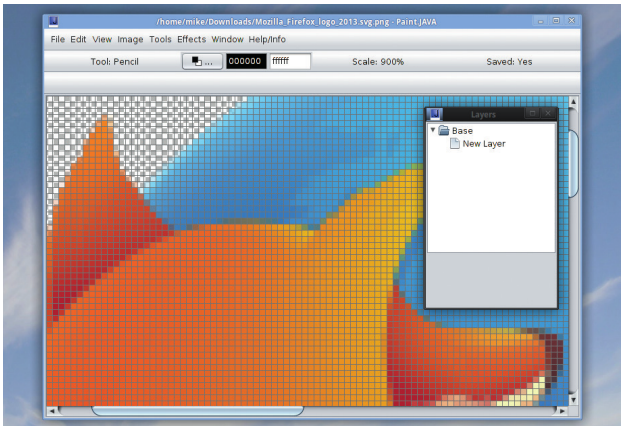
Paint.JAVA was borne of frustration with existing pixel-editing tools. "Pinta is horrible, and Gimp is over-complicated and sucks at pixel art", says the developer. That's fair enough: Gimp is designed to be like Photoshop and focus on filters and effects for photos, rather than on plotting individual dots in images. So if you're designing sprites for video games or icons for programs and you haven't found any useful tools just yet, this could be a big help.

Although Paint.JAVA is still in beta, it's already quite usable. To run it, grab the **.jar** file, make sure you have a JRE (Java Runtime Environment) installed, and enter:

```
java -jar Paint.JAVA.jar
```

Don't expect much from the interface right now – it's very bare, and lacking the icon-laden toolbars you would expect to see in an image editor. Fortunately, there's still plenty of functionality tucked away in the menus, so click around and you'll find pencil and brush tools, shape creation tools, and an eraser. A handful of basic effect filters are included, and you can enable a grid mode, which helps when you're trying to keep various elements of a picture aligned.

Paint.JAVA isn't the prettiest pixel editor out there, but it has all the basics covered and thanks to its Java roots it runs pretty much everywhere. With a few refinements and decent selection of plugins, it'll be a great little app one day.



Enabling a temporary grid helps enormously when you're plotting individual pixels and aligning things.

FOSSPICKS BRAIN RELAXERS

Mascot-happy racing game

SUPERTUXKART 0.8.1

http://supertuxkart.sourceforge.net

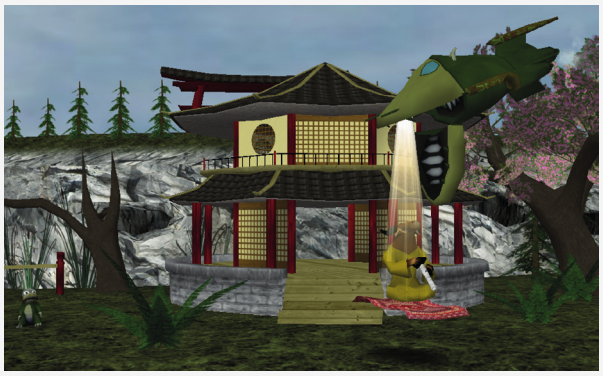
SuperTuxKart has been doing the rounds for many years, so if you haven't played it for a while, you might remember it as a rough-edged Mario Kart clone with bits missing and gameplay flaws. That's certainly how we remembered it, until we tried the latest release. SuperTuxKart has come on in leaps and bounds over the last couple of years, thanks in part to its acceptance in Google's Summer of Code scheme – but also because it has a story mode now.

And this story mode is really well done: the GNU mascot (a gnu, unsurprisingly) gets kidnapped, and your job is to free him by winning races. The level of graphical and audio polish now approaches commercial games, especially during the intro, and the presentation is pretty slick too.

Still game

With 20 courses and 16 playable characters, there's plenty of variety in the game, and many of the courses unlock as you progress through the game. The controls are just about perfect now: the karts react well and don't slip and slide as they did in early releases. If you find the going tough, though, you can try a tutorial mode which takes you through the controls step-by-step.

It's often said that part-time, hobbyist Free Software developers can't produce games that rival commercial counterparts in terms of polish, but SuperTuxKart shows otherwise. It's fun, it's pretty, it's silly, and racing as Tux against the FreeBSD daemon represents what Free Software is all about: different projects often battling each other, but all with the same target in the end.



Yipes! The GNU mascot is being beamed up into a sinister spacecraft. Get your racing gloves on and free him…

Object-hunting fest

E.T. GAME 1.03

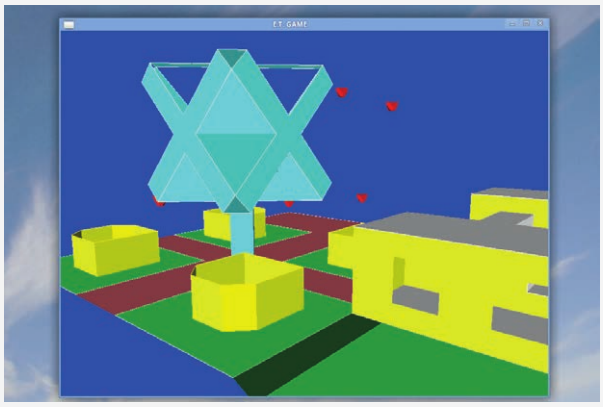
https://github.com/mki1967/et-game

Old games on Linux never die – they just need to be recompiled for the latest distros. Providing that the games are open source, of course. E.T. GAME hasn't changed much since its original release in 2003, but today the code is enjoying retirement in a GitHub repo, receiving the occasional update to make sure that it works with the latest versions of everything.

As you can see from the screenshot, E.T. GAME is totally retro in appearance – and this visual simplicity also means that it has very few dependencies. Providing that you have the development headers for X and GLX installed, you'll be able to get it built with a single **make** command. If the build is successful, enter **sh runme** to start playing.

Your goal is to collect 10 randomly placed red crystals that are scattered around the playing area. This is trickier than it looks, with some crystals tucked away inside rooms and crannies, and once you have them all you need to find a hidden exit. Nine levels are included – and there's even an editor to make your own.

"Old games on Linux never die – they just need to be recompiled"



Use the arrow keys to look around, and F and B to move forward and back. Hold Shift + left or right to sidestep instead of turn.

E.T. GAME won't last you long, but if you've ever fancied doing some 3D programming, the source code is worth exploring for ideas. It demonstrates how to get a simple 3D engine up and running without pulling in shedloads of dependencies, so you can easily modify it to turn it into a first-person shooter or maze exploration romp.