The monthly GNU Column

BRAVE GNU WORLD

Welcome to another issue of Georg CF Greve's Brave GNU World with a mixture of projects that should provide some inspiration for most readers. This month will also begin by introducing some Free Software games

Trophy

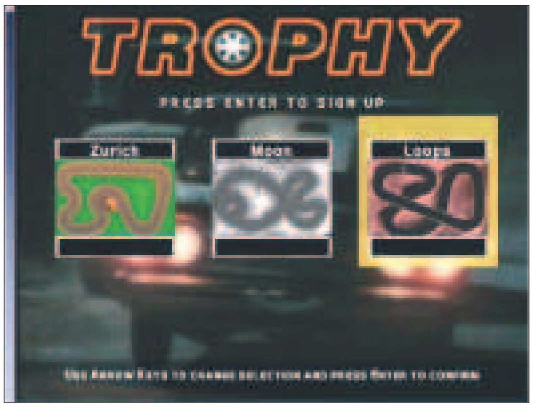
Andrew Mustun began developing Trophy in March 2000 in order to fill the hole that quitting his computer science studies had left. Trophy is a car-racing game in 2D top-view, in which it's not only passing the finish line first which matters. More specifically, it is also about passing the finish line at all, since, in true James Bond style, the cars are armed and more than willing to make use of their gadgets.

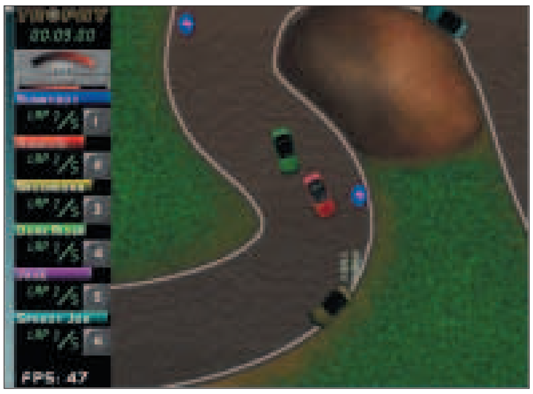
Thanks to solid graphics and sound, Trophy is already a fairly complete game that should allow friends of action-heavy fun to enjoy themselves, although a powerful computer is recommended for smooth scrolling.

Trophy was written in C++ and Andrew releases it under the GNU General Public License. Its biggest weakness is probably the lack of a network play option. Andrew plans to take care of this once he finds time for it again, since right now his newly taken up studies keep him quite busy.

It should be noted that whilst preparing this article, Andrew did emphasise that he was a peace-loving individual who disapproves of guns in general and who also considers cars to be dangerous enough already without heavy weaponry; although as he still lacks a driving license at 25, he couldn't drive a real car anyway.

That shouldn't take the fun out of wasting an entertaining afternoon with Trophy, though. If you seek creative relaxation through designing more tracks with GIMP, that's not a problem. The instructions on how to do so are on the Trophy homepage.


TROPHY
PRESS ENTER TO SIGN UP
Zurich
Moon
Loops
USE ARROW KEYS TO CHANGE SELECTION AND PRESS ENTER TO CONFIRM
Choosing tracks in Trophy



Cornering in Trophy

Should the network option be missed overly much, interested developers are of course welcome to implement it themselves. Andrew would definitely appreciate help in this direction.

Adonthell

The Adonthell project is busy creating a Free 2D role playing game similar to those on the early consoles like the SNES, although it is intended to have a much higher degree of freedom.

Many games consist of two connected parts where each part can be understood individually. The importance of content like story, graphics, music and so on is immediately obvious. The technical component, which does the actual interfacing with the player, for instance, is commonly referred to as the "engine". Usable analogies seem to be the relationship between programs and operating system or scores and musical instruments. This separation allows games to be written in an abstract form with language and possibilities determined by the engine.

The Adonthell project not only develops a game with a fixed story, it also develops an engine for role playing games along with stories that are being

Bathtime in Adonthell

interpreted by this engine. It is planned that all releases will not only contain code fragments but also complete little games.

Taking the usual fluctuations into account, the permanent team of Adonthell are Kai Sterker, Alexandre Courbot, James Nash, Benjamin Walther-Franks, Joel Vennin, Joseph Toscano and Mike Nieforth. Only three of these are programmers, the others are musicians, graphic artists or authors. Thanks to this combination and the fact that all members insist on meeting in person once a year, the game has a professional feel to it and gives a good example of lively cooperation over the Internet.

The code and graphics of Adonthell are released under the GNU General Public License. The programming language used for the engine is C++. For game control, expansions and in-game scripting, the Python interpreter was embedded into the engine. This makes it possible to write games for the Adonthell engine without ever touching C++. For music, OGG Vorbis is being used.

One of the priorities was to be able to package engine and games separately. You only need to install the engine and with some help of tools like map and dialogue editors, as well as a little Python, it is possible to create a complete game. To make this even more interesting, the Adonthell group also works on a complete world with its own history, customs and peculiarities.

Alexandre Courbot, who answered the Brave GNU World questionnaire, said the next step is to rewrite big parts of the code based on experience gathered. The team is also still looking for people willing to help with porting the project to other platforms. The game is known to run on GNU/Linux, FreeBSD, OpenBSD and BeOS, but in the long term, it is also planned to support proprietary operating systems in order to give their users a first taste of freedom.

Internally development is moving towards a more client/server based architecture to allow networked playing. Even though it isn't planned to support very large numbers of players, Adonthell should eventually be capable of allowing a medium number of players to interact in a persistent world.

Since its first steps in 1999, Adonthell has come a fair way already and further development provides hope for more. Those with the right skills who are too impatient to wait are encouraged to help the process along. The Adonthell team welcomes all kind of support. All others can already spend one or two afternoons with "Waste's Edge", the first playable release.

XBindKeys

Philippe Brochard wrote the program XBindKeys, which makes it possible to assign shell commands to keys under X11. This way frequently used programs like the mailer, browser or xterm can be launched by a single key combination without having to take a detour via the menu. Once you get used to being able to do certain things without having to take the hands off the keyboard, you won't want to miss this capability.

Some window managers support keybindings natively, but you'll frequently find that not all of the available keys are freely assignable – like special keys only present on some keyboards. Philippe himself uses XBindKeys to bring up a shutdown menu when pressing the "power" key that his keyboard has.

Everyone who tries out new window managers or

Unhelpful help in Adonthell

[illegible]

Config file for XBindKeys

uses different window managers from time to time knows the problems associated with teaching the new window manager your favourite key combinations. XBindKeys solves both problems, and in combination with the mini-program MoveMouse, also written by Philippe, it can even be used to bring the mouse to previously specified positions with a single key press.

XBindKeys and MoveMouse were both written in C and are available under the GNU General Public License, as Philippe feels them to be his contribution to the GNU System. The configuration of XBindKeys is done by directly editing the easily understood plain ASCII configuration file, or by using the GTK front-end XBindKeys-Config, written by Laurent Vuibert.

As a side note it should be said that one should make sure XBindKeys gets started before the window manager; this ensures key presses reach XBindKeys instead of being intercepted by the window manager.

There are no plans for further development since XBindKeys is stable and Philippe believes it should remain small and useful. This clearly sets a good example in the fight against the spread of "featureism."

Access Road

The area of IT security is without question complex but also important. Awareness of the possibility of bugs in programs is reasonably good by now, but it is also possible to create security problems by combining programs that are working and "bug free."

Complex environments and tasks sometimes require pretty complex and obscure information systems. Without access limitations, data could be abused or modified secretly. The problem of this becomes immediately apparent when thinking about computer-based hospitals, for instance. The opposite, a complete limitation of all access, is also not a good idea since it makes all work impossible. So the task at hand is to design a system in a way that necessary usage is possible, while abuse becomes impossible.

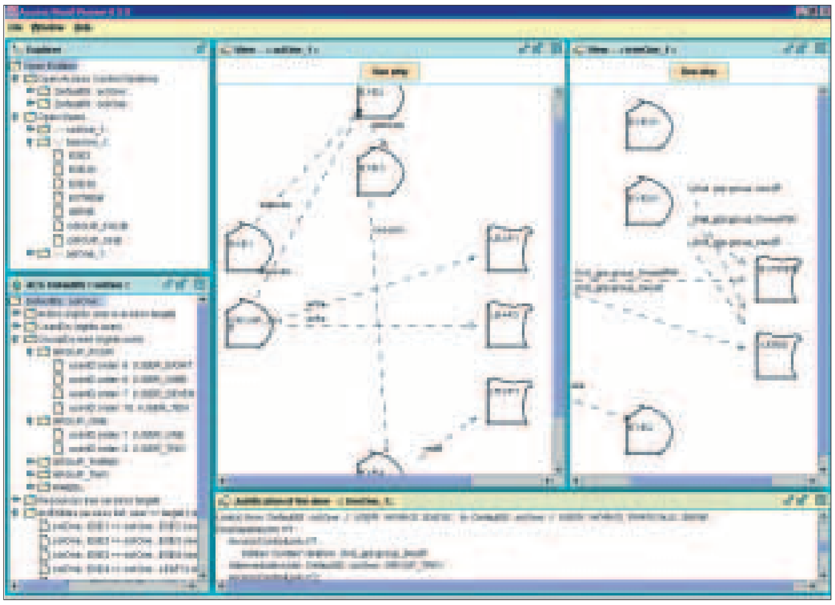
Since the conception of such systems is done by humans there is always a significant potential for error. Access Road by Patrick Thazard tries to model the complex environment of information systems and visualise it for the administrator. This does not increase security in itself, but it offers help in finding security problems that may have gone unnoticed otherwise.

Patrick Thazard himself has been working as a computer security consultant since 1987 and wrote Access Road in Java under the GNU General Public License. The documentation is released under the GNU Free Documentation License.

At the moment Access Road supports GNU/Linux-like systems, simple routers and Solaris-like systems. Even though no data update has been made before the last release, Patrick is convinced it will already provide a very useful structure for other developers.

Further plans including adding the interaction between GNU/Linux and Apache and then adding the interactions between operating system and the employed database management system. In the long-term expansions for modelling systems like CORBA or Windows 2000/XP should complete the system.

Until then there is still quite a bit of work to do and Patrick is looking for other developers willing to support his work. Even if the approach of graphical modelling in general does have some potential weaknesses, this project seems to be quite fascinating and allows exploration in new ways.


Controling access

GNU Cons

GNU Cons offers an alternative to one of the senior programs, GNU Make. Anyone who has ever downloaded software as source code, or recompiled the Linux kernel, has already used GNU Make simply by entering the command *make*.

Make was written in a time when it was still normal to compile programs by hand or with bash shell scripts and at the time it was a big step forward. If you've ever tried writing a makefile yourself, you will most certainly have found the syntax to be pretty hard to understand. This is why most developers copy working makefiles from other projects to modify them for their current purpose, or have makefiles automatically generated by programs like GNU Automake. On top of this, *make* tries to detect modifications by the time of their last modification only, which is problematic and fails for sure when trying to recompile with new compiler options.

The name GNU Cons was chosen because with its help, programs can be "Cons-tructed" and its goal is to do the jobs done by *make* the way they should be done. A very central component for this is a good mechanism to detect whether files have been modified. As a result, GNU Cons does not only look at access times, but also works with MD5 file signatures, allowing a very good means of identification.

A good example of these capabilities is the behaviour of GNU Cons in the following scenario: the program "test" consists of the object files "test1.o", "test2.o" and "test3.o". After the first compilation of the program, the object file "test2.o" is damaged/corrupted deliberately. The next call of GNU Cons will rebuild "test2.o". This will not re-link the program "test" again, since GNU Cons realised that the generated version of "test2.o" is identical to the one previously linked-in.

Should GNU Cons be called with different computer options, all parts are recompiled and linked, since not only the results but also the parameters used to create them are part of the comparison.

GNU Cons was written in Perl by Bob Sidebotham, who has since retired from working on it. Rajesh Vaidheeswarran and Steven Knight took over as maintainers of the stable (Rajesh) and development (Steven) versions. Being part of the GNU Project, GNU Cons is released under the GNU General Public License.

Thanks to the use of Perl, GNU Cons runs on both Windows and Unix-derivates and its configuration files, the equivalent of the commonly used makefiles, allow the use of very dynamic and complex expressions to customise a program to a certain system as much as possible.

GNU Cons is fully usable although it is rather C/C++ heavy. Other programming languages can be accessed with the Cons::Plus module by Johan Holmberg. Also configuration files are of course rather Perl-heavy, which is not exactly a plus for many people.

At the end of this feature I would like to express my thanks to the GNU Cons team for coming up with extremely verbose answers to the standard Brave GNU World questions.

Rajesh Vaidheeswarran and Erich Waelde provided huge amounts of information with many examples, which was very interesting for me even if I was only able to put small parts of it into the issue. I can only hope some of the most important advantages got across well enough to give interested developers an idea of why they ought to give Cons a try.

SCons

The obvious similarities in name between GNU Cons and SCons are not coincidental. The design of SCons is largely based on GNU Cons and was written in August 2000 by Steven Knight, maintainer of the GNU Cons development version, for the Software Carpentry competition.

After being a maintainer of Cons for some time already, he pursued the goal of combining the ideas of Cons with the power of Python for the competition. SCons is based on this work. Like Cons, it provides superior capabilities for detecting modifications and resolving dependencies, but its configuration files are based on Python. Also SCons already supports parallel compilation with the ability to specify the number of threads by a command line option.

SCons is still a fairly young project – the first alpha was released on December 13 2001 – which is why releases are still happening at very short intervals, often containing additional functions. So even though the project has been working with a strong regressive test-infrastructure to maintain backwards-compatibility and quality since day one, one may still experience roughness in everyday use.

The list of planned expansions is still rather long and contains not only good support for Java, C# and Fortran, but also the ability to generate documentation in several formats (PDF, PostScript, TeX and more) and archival.

SCons is being released under an X11-type of license, which is known not to protect the freedoms. It is possible that the GPL would have been the better choice for the long-term perspective, but SCons clearly qualifies as Free Software and taking a look at it or participating is clearly a good idea.

Finish

Alright, so much for the Brave GNU World for this month. For the next issue I already have one project stowed up that I'm already looking forward to introducing, since it has my life much easier and I hope it'll do the same for you.

Until then I wish you all a good time and please don't hold back on suggestions, ideas, comments, feedback and project introductions; preferrably by email.

Info

Send ideas, comments and questions to Brave GNU World	*column@brave-gnu-world.org*
Homepage of the GNU Project	*http://www.gnu.org*
Homepage of Georg's Brave GNU World	*http://brave-gnu-world.org*
"We run GNU" initiative	*http://www.gnu.org/brave-gnu-world/rungnu/rungnu.en.html*
Trophy homepage	*http://trophy.sourceforge.net*
Adonthell homepage	*http://adonthell.linuxgames.com*
OGG Vorbis homepage	*http://www.xiph.org/ogg/vorbis*
Join Adonthell	*http://adonthell.linuxgames.com/development/join.shtml*
Xbindkeys homepage	*http://hocwp.free.fr/xbindkeys/xbindkeys.html*
MoveMouse homepage	*http://hocwp.free.fr/movemouse.html*
Xbindkeys-Config homepage	*http://www.netchampagne.com/xbindkeys_config*
Access Road homepage	*http://accessroad.sourceforge.net*
GNU Cons homepage	*http://www.gnu.org/software/cons*
GNU Make homepage	*http://www.gnu.org/software/make*
SCons homepage	*http://www.scons.org*