The Linux Gamers' HOWTO

Peter Jay Salzman

Frédéric Delanoy

2004−11−13 v.1.0.6

Abstract

The same questions get asked repeatedly on Linux related mailing lists and news groups. Many of them arise because people don't know as much as they should about how things "work" on Linux, at least, as far as games go. Gaming can be a tough pursuit; it requires knowledge from an incredibly vast range of topics from compilers to libraries to system administration to networking to XFree86 administration ... you get the picture. Every aspect of your computer plays a role in gaming. It's a demanding topic, but this fact is shadowed by the primary goal of gaming: to have fun and blow off some steam.

This document is a stepping stone to get the most common problems resolved and to give people the knowledge to begin thinking intelligently about what is going on with their games. Just as with anything else on Linux, you need to know a little more about what's going on behind the scenes with your system to be able to keep your games healthy or to diagnose and fix them when they're not.

Table of Contents

Table of Contents

Table of Contents

1. Administra

If you have ideas, corrections or questions relating to this HOWTO, please email me. By receiving feedback on this howto (even if I don't have the time to answer), you make me feel like I'm doing something useful. In turn, it motivates me to write more and add to this document. You can reach me at <p(at)dirac(dot)org>. My web page is http://www.dirac.org/p and my Linux pages are at http://www.dirac.org/linux. Please do send comments and suggestions for this howto. Even if I don't take your suggestions, your input is graciously received.

I assume a working knowledge of Linux, so I use some topics like runlevels and modules without defining them. If there are enough questions (or even protests) I'll add more basic information to this document.

1.1. Authorship and Copyright

If you want to create a derivative work or publish this HOWTO for commercial purposes, I would appreciate it if you contact me first. This will give me a chance to give you the most recent version. I'd also appreciate either a copy of whatever it is you're doing or a spinach, garlic, mushroom, feta cheese and artichoke heart pizza.

1.2. Acknowledgements

Thanks goes out to these people for extensive comments, corrections, and diffs. Their effort is above and beyond the call of duty:

Frédéric Delanoy, Moritz Muehlenhoff <jmm(at)Informatik(dot)uni−bremen(dot)de>, Mike Phillips, Ioan Rogers <buck(at)aiur(dot)co(dot)uk>

I would also like to thank the following people for sending in comments and corrections. Without their help, there would be more typos and mistakes than you could shake a stick at:

Michael McDonnell

1.3. Latest Versions and Translations

The latest version can be found at http://cvs.sourceforge.net/cgi−bin/viewcvs.cgi/lgh/LG−HOWTO or http://www.dirac.org/linux/writing, but this is my own personal working copy. The version at my personal web site might be broken if I'm working on the HOWTO. The version at sourceforge is bleeding edge but guaranteed to be not broken, however it may have glitches, like unfinished paragraphs. :)

The most recent stable version can be found at http://www.tldp.org.

1.3.1. Russian

Dmitry Samoyloff <dsamoyloff(at)yandex(dot)ru> is the maintainer of the Russian translation. The most recent version can be found at http://www.dirac.org/linux/writing.

1.3.2. Hungarian

László Daczi <dacas(at)korhaz(dot)rethy(dot)hu>, the Hungarian LDP coordinator, announced that a Hungarian translation was produced by Szilard Ivan, and is available at http://tldp.fsf.hu/HOWTO/Linux−Gamers−HOWTO−hu.

2. Definitions: Types Of Games

Not everyone knows the different types of games that are out there, so in an effort to form a common language that we can all use, I'll run through each game type and provide a very brief history.

2.1. Arcade style

Although arcade games had their heydey in the 80's, they are nonetheless very popular. Nothing will ever replace walking into a dark, crowded and noisy arcade gallery, popping a quarter into your favorite machine and playing an old fashioned game of Space Invaders. Arcade style games attempt to simulate the arcade games themselves. There is such a vast number of these things that it's nearly impossible to enumerate them all, but they include clones of Asteroids, Space Invaders, Pac−Man, Missile Command and Galaxian.

2.2. Card, logic and board games

Computer based card games simulate a card game like poker or solitaire. The program can simulate your opponent(s).

Logic games usually simulate some well known logic puzzle like Master Mind or the game where you have put sliding numbered tiles in order inside a box.

Computer based board games simulate some kind of board game you'd play on a table top with friends, like monopoly, Mille Bourne, chess or checkers. The program can simulate your opponent.

2.3. Text Adventure (aka Interactive Fiction)

Once upon a time, when Apple][, Commodore, and Atari ruled the world, text adventures were the game of choice of `intelligent folk'. You are given a scenario and can interact with the world you're placed in:

```
You are in a room.  It is pitch dark and you're likely to be eaten by a grue.
> Light lantern with match.
You light the lantern.  This room appears to be a kitchen.  There's a table with a
book in the center.  You also see an oven, refrigerator and a door leading east.
> Open the oven.
In the oven you see a brown paper bag.
> Take the bag.  Open the bag.  Close the oven.
Inside the bag is a some garlic and a cheese sandwich.  The oven door is now closed.
```

Back then, text adventures were self contained executables on a disk or casette. These days there's usually a data file and an interpreter. The interpreter reads data files and provides the gaming interface. The data files are the actual game itself, similar to the relationship between first person shooters (Section 2.7) and `wad` files.

The first adventure game was Adventure (actually "ADVENT", written on a PDP−1 in 1972). You can play Adventure yourself (actually, a descendent); it comes with "bsd games" on most Linux distros. Text adventures became popularized by Scott Adams (Section 11.5) and reached their height of popularity in the late 80's with Infocom (Section 11.4) which are also playable under Linux.

As computer graphics became easier and more powerful, text adventures gave rise to graphic adventures. The death of commercial interactive fiction more or less coincided with the bankruptcy of Infocom.

2.4. Graphical Adventures

Graphical adventures are, at heart, text adventures on steroids. The degree to which they use graphics varies widely. Back in the 80's, they were little more than text adventures which showed a screen of static graphics. When you picked up an item, the background would be redrawn without the item appearing. The canonical example would be the so−called `Hi−Res Adventures' like The Wizard And The Princess. Later on, the sophisticated graphical adventures had your character roaming around the screen, and you could even use a mouse, but the interface remained purely text.

Next there are the `point and click adventures' which basically have no text interface at all, and often have dynamic graphics, like a cat wandering around the room while you're deciding what to do next. In these games, you point at an object (say, a book) and can choose from a pull−down list of functions. Kind of like object oriented adventuring. :) There aren't many graphical adventures written natively for Linux. The only one I can think of is Hopkins FBI (which happens to be my favorite game for Linux).

2.5. Simulation (aka Sims)

Simulations strive to immerse the player behind the controls of something they normally wouldn't have access to. This could be something real like a fighter jet or something imaginary like a mechanized warrior combat unit. In either case, sims strive for realism.

Some sims have little or no strategy. They simply put you in a cockpit to give you the thrill of piloting a plane. Some are considerably complex, and there's often a fine line between sims and strats (Section 2.6). A good example would be Heavy Gear III or Flight Gear. These days sims and strats are nearly indistinguishable, but a long time ago, sims were real time while strats were turn based. This is awkward for modern day use, since a game like Warcraft which everyone knows as a strat, would be a sim by definition.

2.6. Strategy (aka Strats)

Strategy games have their roots in old Avalon type board games like Panzer Leader and old war strategy games published by SSI. Generally, they simulate some kind of scenario. The scenario can be peaceful, like running a successful city (SimCity), or not, like illegal drug selling operation (DrugWars) or an all−out war strategy game like Myth II. The types of games usually take a long time to complete and require a lot of brainpower.

Strats can be further divided into two classes: real time and turn based. Real time strats are based on the concept of you−snooze−you−lose. For example, you're managing a city and a fire erupts somewhere. The more time it takes for you mobilize the fire fighters, the more damage the fire does. Turn based strats are more like chess−−−the computer takes a turn and then the player takes a turn.

2.7. First Person Shooter (aka FPS)

What light through yonder window breaks? It must be the flash of the double barreled shotgun! We have a long and twisted history with FPS games which started when id Software open sourced code for Doom. The code base has forked and merged numerous times. Other previously closed engines opened up, many engines are playable via emulators, many commercial FPS games were released for Linux and there are quite a number of FPS engines which started life as open source projects. Although you may not be able to play your *favorite* FPS under Linux (Half−Life plays great under winex) Linux definitely has no deficiency here!

First person shooters are characterized by two things. First, you pretty much blow up everything you see. Second, the action takes place in first person. That is, through the eyes of the character who's doing all the shooting. You may even see your hands or weapon at the bottom of the screen. They can be set in fantasy (Hexen), science fiction (Quake II), present day `real world' (Soldier Of Fortune) and many other settings.

Like text adventures, FPS fit the engine/datafile format. The engine refers to the actual game itself (Doom, Quake, Heretic2) and plays out the maps and bad guys outlined by the datafile (`doom2.wad`, `pak0.pak`, etc). Many FPS games allow people to write their own non−commercial datafile. There are hundreds, even thousands of non−commercial Doom datafiles that you can download for free off the net. Often, companies release their engines to the open source community so we can hack and improve them. However, the original data files are kept proprietary. To this day, you still have to purchase `doom.wad`.

2.8. Side Scrollers

Side scrollers are similar to FPS but you view your character as a 2D figure who runs around various screens shooting at things or performing tasks. Examples would be Abuse for Linux and the original Duke Nukem. They don't necessarily have to be violent, like xscavenger, a clone of the old 8−bit game Lode Runner.

2.9. Third Person Shooters

Similar to FPS, but you view your character in third person and in 3D. On modern third person shooters you can usually do some really kick−butt maneuvers like Jackie Chan style back flips and side rolls. The canonical example would be Tomb Raider. On the Linux platform, we have Heretic 2 and Heavy Metal FAKK2.

2.10. Role Playing Game (aka RPG)

Anyone who has played games like Dungeons & Dragons or Call of Cthulhu knows exactly what an RPG is. You play a character, sometimes more than one, characterized by traits (eg strength, dexterity), skills (eg explosives, basket weaving, mechanics) and properties (levels, cash). As you play, the character becomes more powerful and the game adjusts itself accordingly, so instead of fighting orcs, at high levels you start fighting black dragons. The rewards increase correspondingly. At low levels you might get some gold pieces as a reward for winning a battle. At high levels, you might get a magic sword or a kick−butt assault rifle.

RPG's generally have a quest with a well defined ending. In nethack you need to retrieve the amulet of Yendor for your god. In Ultima II, you destroy the evil sorceress Minax. At some point, your character becomes powerful enough that you can `go for it' and try to complete the quest.

While the insanely popular Ultima series, written by Richard Garriot (aka Lord British) for Origin, was not the first RPG, it popularized and propelled the RPG genre into mainstream. Ultima I was released in 1987 and was the game that launched 9 (depending on how you want to count them) very popular sequels, finishing with Ultima IX: Ascension. You can play Ultima VII under Linux with Exult (Section 11.7).

The canonical RPG on Linux is Rogue (the ncurses library started life as a screen handling routine for Rogue!) and it has infinite variants like Zangband and Nethack (which has many variants itself). Some RPG's are quite complicated and great feats of programming. There seems to be a deficiency of commercial RPGs for Linux. Not counting the rogue variants, there's also a deficiency of open source RPGs too.

3. Libraries

We'll run through the different gaming libraries you'll see under Linux.

3.1. What is Glide2?

Glide2 is a low level graphics API and driver that accesses 3D hardware accelerated functions on 3dfx's Voodoo I, II and III cards, under XFree86 3.x.

A program can only use the special hardware accelerated features of these cards by using the Glide2 library in one of two ways:

- directly written using Glide2 (Myth II, Descent III)
- indirectly using Mesa built with a Glide2 backend to simulate OpenGL (Rune, Unreal Tournament)

3dfx opened up the specifications and source code to the open source community. This allowed Daryll Strauss to port Glide2 to Linux which enabled XFree86 3.x users to use Voodoo I, II and III cards under Linux.

Since Glide2 accesses the video card directly, Glide2 applications will either need to be run by root or be setuid root. A way around this was to create the kernel 3dfx module. This module (and its device file `/dev/3dfx`) allows Glide2 graphical hardware acceleration for non−root users of non−setuid applications.

Unfortunately, Glide2 is also a dead issue. It's only used for Voodoo I, II, III boards (which are becoming outdated), under XFree86 3.x (most people use XFree86 4.x). And since 3dfx is now a defunct company, it's a sure bet that no more work will be done on Glide2 and no more games will be written using Glide2.

3.2. What is Glide3?

Unlike Glide2, Glide3 is not an API used for game programming. It exists only to support DRI on Voodoo III, IV and V boards under XFree86 4.x. None of the games which use Glide2 will work with Glide3. This shouldn't be a surprise since Glide2 and Glide3 support different video cards and different versions of XFree86. The only video card that can use both Glide2 (under XFree86 3.x) and Glide3 (under XFree86 4.x) is the Voodoo III. It's reported that a Voodoo III using Glide2 will outperform a Voodoo III using Glide3.

When you use a Voodoo III, IV or V under XFree86 4.x, you want to use a version of Mesa (see Section 3.4) which was compiled to use Glide3 as a backend to ensure hardware accelerated OpenGL on your system.

3.3. What is OpenGL?

OpenGL is a high level graphics programming API originally developed by SGI, and it became an industry standard for 2D and 3D graphics programming. It's defined and maintained by the Architectural Revision Board (ARB), an organization which include representatives from SGI, IBM, DEC, and Microsoft. OpenGL provides a powerful, complete and generic feature set for 2D and 3D graphics operations.

There are 3 canonical parts to OpenGL:

- GL: The OpenGL core calls
- GLU: The utility calls

* GLUT: OS independent window event (mouse, keyboard, etc.) handler.

OpenGL is not only an API, it's also an implementation, written by SGI. The implementation tries to use hardware acceleration for various graphics operations whenever available, which depends on what videocard you have in you computer. If hardware acceleration is not possible for a specific task, OpenGL falls back on software rendering. This means that when you get OpenGL from SGI, if you want any kind of hardware acceleration at all, it must be OpenGL written and compiled specifically for some graphics card. Otherwise, all you'll get is software rendering. The same thing is true for OpenGL clones, like Mesa.

OpenGL is the open source equivalent to Direct3D, a component of DirectX (Section 3.14). The important difference being that since OpenGL is open (and DirectX is closed), games written in OpenGL are much easier to port to and co−develop on Linux than games written using DirectX.

3.4. What is Mesa?

Mesa <http://www.mesa3d.org> is a free implementation of the OpenGL API, designed and written by Brian Paul. While it's not officially certified (that would take more money than an open source project has), it's an almost fully compliant OpenGL implementation conforming to the ARB specifications. It's reported that Mesa is even faster than SGI's own OpenGL implementation.

Just like OpenGL, Mesa makes use of hardware acceleration whenever possible. When a particular graphics task isn't able to be hardware accelerated by the video card, it's software rendered; the task is done by your computer's CPU instead. This means that there are different builds of Mesa depending on what kind of video card you have. Each build uses a different library as a backend renderer. For example, if you have a Voodoo I, II or III card under XFree86 3.x, you'd use mesa+glide2 (written by David Bucciarelli) which is the Mesa implementation of OpenGL that uses Glide2 as a backend to render for graphical operations.

3.5. What is DRI?

Graphics rendering has 3 players: the client application (like Quake 3), the X server and the hardware (the graphics card). Previously, client applications were prohibited from writing directly to hardware, and there was a good reason for this. A program that is allowed to directly write to hardware can crash the system in any number of ways. Rather than trusting programmers to write totally bug free, cooperative programs that access hardware, Linux simply disallowed it. However, that changed under X 4.x with DRI (Direct Rendering Infrastructure <http://www.dri.sourceforge.net>. DRI allows X clients to write 3D rendering information directly to the video card in a safe and cooperative manner.

DRI gets the X server out of the way so the 3D driver (Mesa or OpenGL) can talk directly to the hardware. This speeds things up. The 3D rendering information doesn't even have to be hardware accelerated. On a technical note, this has a number of virtues.

* Vertex data doesn't have to be encoded/decoded via GLX.
* Graphics data isn't sent over a socket to the X server.
* On uni−processor machines the CPU doesn't have to change context between XFree86 and its client to render the graphics.

3.6. What is GLX?

GLX is the X extension used by OpenGL programs, it is the glue between the platform independent OpenGL and platform dependent X.

3.7. What is Utah GLX?

Utah−GLX is the precursor to DRI. It makes some different design decisions regarding separation of data and methods of accessing the video card like relying on root access rather than creating the kernel infrastructure for secure access. It provides support for a few cards which are not well supported by DRI like the ATI Rage Pro family, S3 Virge (although anyone using this for gaming is, well, nuts), and an open source TNT/TNT2 driver (which is very incomplete). The TNT/TNT2 driver is based on reverse−engineering of the obfuscated source code release of the X 3.3 drivers by nVidia. However, they're really incomplete, and effectively, unusable.

3.8. What is xlib?

Every once in awhile you'll see some sicko (said with respect) write a game in xlib. It is a set of C libraries which comprise the lowest level programming interface for XFree86. Any graphics programming in X ultimately makes use of the xlib library.

It's not an understatement to say that xlib is long winded, arcane and complicated. Because of this, there are lots of libraries like SDL (Section 3.10) for 2D graphics, OpenGL (Section 3.3) for 3D graphics and widget sets (Section 3.9) for widgets within windows which hide the details of different aspects of xlib programming.

While some games are written in xlib, like the Doom Editor Yadex, xlib itself is not a serious game writing library. Most games don't need the low−level interface that xlib provides. In addition, by using the higher level libraries, a game writer can develop his game on multiple platforms, even ones that don't use XFree86.

3.9. What is a widget set?

Widgets are objects that make up a GUI application's interface. They include things like text entry boxes, pulldown menus, slider bars, radio buttons and much more. A widget set is a collection of related widgets that are designed to have a common interface and a consistant "feel". Gtk is the canonical widget set on Linux, but there are many others like fltk (a small C++ widget set), Xaw, Qt (the widget set of KDE), and Motif (the widget set used by Netscape). Motif used to be the king of widget sets in the Unix world, but it was very expensive to license. The Open Group finally opened up Motif's license for open source operating systems, but it was too little too late. There are many completely open source widget sets which are more complete and much nicer looking than Motif, including Lesstif, a totally free Motif clone.

3.10. What is SDL (Simple DirectMedia Layer)?

SDL <http://www.libsdl.org> is a library by Sam Lantiga (graduate of UCD, yeah!). It's actually a meta−library, meaning that not only is it a graphics library which hides the details of xlib programming, it provides an easy interface for sound, music and event handling. It's LGPL'd and provides joystick and OpenGL support as well. Unlike xlib (Section 3.8), SDL is very suited for game programming.

The most striking part of SDL is that it's a cross platform library. Except for a few details, a program written in SDL will compile under Linux, MS Windows, BeOS, MacOS, MacOS X, Solaris, IRIX, FreeBSD, QNX and OSF. There are SDL extensions written by various people to do things like handle any graphics format you care to mention, play mpegs, display truetype fonts, sprite handling and just about everything under the sun. SDL is an example of what all graphics libraries should strive for.

Sam had an ulterior motive for writing such a cool library. He was the lead programmer for Loki Software (he now codes for Blizzard Software), which used SDL in all of its games except for Quake3.

3.11. What is GGI?

GGI <http://www.ggi−project.org> is a project which aims to implement a graphics abstraction layer in lower level code, put graphics hardware support into a common codebase, and bring higher stability and portability to graphics applications. LibGGI applications run on SVGAlib, fb, and X servers among others. Judging from their screenshots, this is quite a powerful library.

Applications that use LibGGI directly include Heroes, Ultrapoint, Quake, and Berlin. Most applications that use SVGALib can be run on X or any other LibGGI backend by using a wrapper library which re−implements SVGALib (Section 3.12) using LibGGI. SDL (Section 3.10) and clanlib (Section 3.15) applications can display on LibGGI but often the native drivers for these libraries are faster, however it's a good way to get SDL, clanlib, and SVGALib applications to run where they would not before.

GGI has a sister project, KGI, which is developing a kernel−level alternative to systems like the linux framebuffer and the DRI. This project is much less far along than LibGGI itself, but promises to combine DRI−level speeds and the stability and security UNIX users expect.

3.12. What is SVGAlib? Frame buffer? Console?

The console is the dark non−graphical screen you look at when your computer first boots up (and you don't have xdm or gdm running). This is opposed to the X environment which has all sorts of GUI things like xterms. It's a common misconception that X means graphics and console means no graphics. There are certainly graphics on the console we will discuss the two most common ways to achieve this.

SVGAlib is a graphics library that lets you draw graphics on the console. There are many graphical applications and games that use SVGAlib like zgv (a console graphical image viewer), prboom and hhexen. I happen to be a fan of this library and of graphical console games in general; they are extremely fast, fullscreen and compelling. There are three downsides to SVGAlib. First, SVGAlib executables need to be run by root or be setuid root, however, the library releases root status immediately after the executable begins to run. Secondly, SVGAlib is video card dependent−if your video card isn't supported by SVGAlib, you're out of luck. Third, SVGAlib is Linux only. Games written in SVGAlib will only work on Linux.

Frame buffers are consoles implemented by a graphics mode rather than a BIOS text mode. Why simulate text mode in a graphical environment? This allows us to run graphical things in console, like allowing us to choose any font we want the console to display (which is normally set by BIOS). There's a good Framebuffer HOWTO available from LDP. Graphical console games written using the frame buffer suffer from the same deficiencies of the SVGA library: not all hardware is supported and the code will only run on Linux.

3.13. What is OpenAL?

OpenAL <http://www.openal.org> aims to be for sound what OpenGL is for graphics. It started as a joint project between Loki Software and Creative Labs, setting out to be a vendor neutral and cross platform API for audio − the audio equivalent of OpenGL (Section 3.3). Loki is no longer in business, but Creative and the Open Source community have kept the project alive. It is licensed LGPL and the specs can be obtained for free from the OpenAL website. It has support from nVidia (nForce2/3 based motherboards come with OpenAL MS Windows libraries for the on−board audio), Apple has added OpenAL to their audio framework for OSX and it can also be found powering the Epic Games Unreal Engine

Currently, it's not all cross−platform goodness. There is almost no support for enhancements like EAX or any hardware acceleration on Linux, though it does it exist in the Windows implementation. However, if you have a Creative SoundBlaster or Audigy sound card (with an emu10x chip), and you use ALSA sound drivers, you can get OpenAL libraries from http://www.lost.org.uk that provide hardware acceleration and decent surround support.

3.14. What is DirectX?

DirectX is a collection of proprietary multimedia API's, first developed by Microsoft in 1995, for its various Windows OS's. It's a mistake to say something like "DirectX is like OpenGL" or "DirectX is like SDL", as is commonly said in DirectX tutorials. Multimedia API's are more centralized on Windows than they are on Linux. A more accurate statement would be something like "DirectX is like DRI, OpenGL and SDL combined". As of October 2004, the most recent version of DirectX is 9c. The components of DirectX are:

DirectDraw
: DirectDraw gives direct access to video memory, like DRI, so 2D graphics can be blitted directly to the video card. DirectDraw is like the graphical component of SDL, but the direct video card access is done by DRI rather than SDL. This is why a game can easily take out a Windows system but should not take down a Linux system.

Direct3D (D3D)
: Direct3D, like OpenGL, provides a 3D graphics API. Whereas OpenGL is open source, lower level and compiles under a multitude of operating systems, D3D is proprietary, higher level and only compiles on Windows. D3D first appeared in DirectX 2, released in 1996.

DirectAudio
: DirectAudio is a combination of 2 audio API's, DirectSound and DirectMusic, which allows direct access to the sound card for sound and music playback.

DirectInput
: DirectInput gives support for gaming input devices such as joysticks.

DirectPlay
: DirectPlay gives support for simplified networking for multiplayer gaming.

DirectShow
: DirectShow provides support for movie files like AVI and MPG. It was a separate API from DirectX, but was integrated with DirectX 8.

DirectSetup
: This API provides a way to install DirectX from within an application to simplify game installation.

Depending on the version of DirectX you're talking about, DirectX support in winex (Section 10.5.3) ranges from well supported to "kind of" supported. It's poorly supported by wine (Section 10.5.1), barely supported by vmware (Section 10.5.5) and unsupported by Win4Lin (Section 10.5.4).

One comment about portability: Each component of DirectX has multiple corresponding library on Linux. Moreover, a game writer who uses libraries like OpenGL, GGI or SDL will write a game which will trivially compile on Windows, Linux and a multitude of other OS's. Yet game companies persist using DirectX and therefore limit their audience to Windows users only. If you're a game writer, please consider using cross platform libraries and stay away from DirectX.

A company named realtechVR started an open source project, DirectX Port, <http://www.v3x.net/directx> which, like wine, provides a Direct3D emulation layer that implements Direct3D calls. The project was focused on the BeOS platform, but is now focused on MacOS and Linux. You can get the latest cvs from their sourceforge page at <http://sourceforge.net/projects/dxglwrap>.

3.15. Clanlib

ClanLib is a medium level development kit. At its lowest level, it provides a platform independent (as much as that is possible in C++) way of dealing with display, sound, input, networking, files, threading and such. ClanLib builds a generic game development framework, giving you easy handling of resources, network object replication, graphical user interfaces (GUI) with theme support, game scripting and more.

4. XFree86 and You

If you're going to game under X, it's crucial that you know a bit about X. The "X Window User HOWTO", and especially "man XF86Config" are *required* reading. Don't short change yourself; read them. They have an extremely high "information to space" ratio. Many problems can be fixed easily if you know your way around `XF86Config` (or `XF86Config-4`).

4.1. Getting information about your X system

Whether you're trying to diagnose an X problem or requesting help from a mailing list or Usenet newsgroup, you'll want to have as much information available as possible. These are a set of tools you can use to obtain that information.

4.1.1. Probeonly

One of the best diagnostic tools and sources of information about your X system is **probeonly** output. To use it, kill X if it's already running and from a console, type:

```
X -probeonly 2> X.out
```

Yes, that's a single dash; so much for standards. The output of X goes to stderr, so we have to redirect stderr with "2>" to a file named `X.out`. This file will have almost everything there is to know about your X system. It's crucial that you know the difference between the various markers you'll see in probeonly output:

```
(--) probed               (**) from config file    (==) default setting
(++) from command line    (!!) notice              (II) informational
(WW) warning              (EE) error               (??) unknown.
```

Here's an example of some information I gleaned from my output:

I'm running at 16 bpp color:

```
(**) TDFX(0): Depth 16, (--) framebuffer bpp 16
```

X has detected what my videocard chipset and videoram are:

```
(--) Chipset 3dfx Voodoo5 found
(--) TDFX(0): VideoRAM: 32768 kByte Mapping 65536 kByte
```

4.1.2. Getting info about your setup: xvidtune

xvidtune is your friend when your X screen is shifted a little bit too far to the right, or if the vertical length is too small to fit on your monitor. However, it's a great diagnostic tool also. It'll give you:

- the hsync/vsync range specified in your XF86Config file

- the 4 horizontal and 4 vertical numbers which defines your videomode (the 1st horizontal/vertical numbers gives the screen resolution). These 8 numbers will tell you which modeline your X uses. See the XFree86 Video Timings Howto for more information. Note that explicit modelines are no longer necessary, since XFree 4.0.1 and up computes modetimings automatically based on your monitor's and video card's capabilities. However, there may be times when you'll want to play around with mode timings, like for weird hardware or if want to tweak your display.
- the "dot clock" your videocard is running at.

4.1.3. Getting info about your setup: xwininfo

xwininfo tells you all sorts of information about X windows. And actually, your "background" or "root" window is considered a window too. So when xwininfo asks you to click on the window you want the information on, click on your background. It'll tell you things like screen and window resolution, color depth, window gravity state (which gives a hint to the window manager about where to place new windows), backing store usage and more.

4.1.4. Other sources of information

xdpyinfo gives cool stuff, like X version and loaded extensions (invaluable when trying to see what's missing, like GLX, DRI, XFree86−VidMode, etc.).

4.1.5. Getting information about your 3D system

glxinfo gives lots of useful information about OpenGL like whether direct rendering enabled, the currently installed versions of glx and mesa, vendor/renderer strings, the GL library files being used and more.

4.2. Playing Games In X Without a Window Manager

When playing a game under X, you should consider starting X without a window manager (WM). Heavyweight WMs, like Enlightenment, or full−blown desktop environments like GNOME or KDE, may produce a noticeable slow down. Even lightweight WMs, like twm, rob your CPU of clock cycles (and in twm's case, even full screen games will have a frame around the window). Running a game without a WM or DE depends on how you access X. If you usually log in to a Virtual Console and start X with "startx" try the following:

Modify `~/.xinitrc`, which tells X what to run upon starting. Here is what my .xinitrc looks like:

```
#quake3 +set r_gldriver libGR.so.1
#exec ut
#lsdldoom −server 2
#exec tribes2
exec /usr/bin/enlightenment
```

You'll usually see a window or desktop manager being executed from this file (GNOME or KDE). Comment out the lines containing the WM or desktop manager with a pound sign (#) and place your game on a new line with any command line arguments you want to pass. If the game is not located in your $PATH, give its full path name.

If you log directly into X using gdm, then things are a little different. These instructions are for gdm 2.4 or greater. They *may* work with kde, but I cannot say for certain.

First, check your `gdm.conf` (usually in `/etc/X11/gdm` or `/etc/gdm`) file for a line that says begins "*SessionDesktopDir=blah*". One of the directories listed as options should be "`/usr/share/xsessions`", and is the directory which will be used in this example. As root, change to the "`/usr/share/xsessions`" directory and take a look at its contents. It should contain some `.desktop` files, each corresponding to an entry you'll see in gdm's Session menu, e.g `gnome.desktop`, `enlightenment.destop`. This example will show you how to log in to Doom3. Copy any of the desktop files to "`doom3.desktop`" and open the new file in your favourite text editor. The file will be full of alternative languages, so cut out everything you don't want and make the file look like this:

```
[Desktop Entry]
Encoding=UTF-8
Name=DOOM III
Comment=iD's Doom III
#if game is not in path, remember to put the full path here
Exec=/usr/games/doom3/doom3
# no icon yet, only the top three are currently used
Icon=
Type=Application
```

Save the file and log out of your window manager. At the gdm login screen, you should now see "*DOOM III*" as an option in "Sessions". Naturally you can add a .desktop file for each game you have installed

5. Various Topics

5.1. Memory Type Range Registers

Starting with Pentium class processors and including Athlon, K6−2 and other CPUs, there are Memory Type Range Registers (MTRR) which control how the processor accesses ranges of memory locations. Basically, it turns many smaller separate writes to the video card into a single write (a burst). This increases efficiency in writing to the video card and can speed up your graphics by 250% or more.

See `/usr/src/linux/Documentation/mtrr.txt` for details. Note that since this file was written, XFree86 has been patched to automatically detect your video RAM base address and size and set up the MTRRs.

5.2. Milking performance from your system for all it's worth

- If for some reason you're using X 3.3, follow the instructions given by `mtrr.txt` (see Section 5.1) to set up your MTRRs. X 4.0 does this automatically for you.
- If you're playing a game under X, don't run a window manager, and *certainly* don't run a desktop manager like GNOME or KDE. See Section 4.2 for details.

  Kill all non−essential processes (you'll have to do this as root) by using the startup scripts on your system. On Debian, the startup scripts for run−level 2 are located in /etc/rc2.d/. You can kill a service in an orderly manner by sending its startup script the `stop' command:

  ```
  # cd /etc/rc2.d
  # ./ntpd stop
  ```

  Another (radical) option is to simply put yourself in single−user mode with

  ```
  # telinit 1
  ```

  This will even get rid of getty; your system will only be running whatever is absolutely crucial to its operation. You'll have something like 10 processes running. The downside is that you'll have to play the game as root. But your process table will be a ghost town, and all that extra CPU will go straight to your game.

5.3. About libraries on Linux

A common problem you'll see in gaming is a library file not being found. They're kind of mysterious and have funny names, so we'll go over libraries on Linux for a bit. There are two types of libraries, static and dynamic. When you compile a program, by default, **gcc** uses dynamic libraries, but you can make **gcc** use static libraries instead by using the *−static* switch. Unless you plan on compiling your games from source code, you'll mainly be interested in dynamic libraries.

5.3.1. Dynamic libraries

Dynamic libraries, also called a "shared library", provide object code for an application while it's running. That is, code gets linked into the executable at run time, as opposed to compile time. They're analagous to the *.dll*'s used by Windows. The program responsible for linking code "on the fly" is called **/etc/ld.so**, and the dynamic libraries themselves usually end with *.so* with a version number, like:

```
/usr/lib/libSDL.so
/lib/libm.so.3
```

When using **gcc**, you refer to these libraries by shaving off the strings *lib*, *.so* and all version numbers. So to use these two libraries, you would pass **gcc** the *−lSDL −lm* options. **gcc** will then `place a memo inside the executable' that says to look at the files `/usr/lib/libSDL.so` and `/lib/libm.so.3` whenever an SDL or math function is used.

5.3.2. Static libraries

In contrast to dynamic libraries which provide code while the application runs, static libraries contain code which gets linked (inserted) into the program while it's being compiled. No code gets inserted at run time; the code is completely self−contained. Static libraries usually end with *.a* followed by a version number, like:

```
/usr/lib/libSDL.a
/usr/lib/libm.a
```

The *.a* files are really an archive of a bunch of *.o* (object) files archived together, similar to a tar file. You can use the **nm** to see what functions a static library contains:

```
% nm /usr/lib/libm.a
...
e_atan2.o:
00000000 T __ieee754_atan2

e_atanh.o:
00000000 T __ieee754_atanh
00000000 r half
00000010 r limit
00000018 r ln2_2
...
```

When using **gcc**, you refer to these libraries by shaving off the strings "lib", ".a" and all version numbers. So to use these two libraries, you would pass **gcc** the *−lSDL −lm* options. **gcc** will then `bolt on' code from `/usr/lib/SDL.a` and `/usr/lib/libm.a` whenever it sees a math function during the compilation process.

5.3.3. How are library files found

If you compile your own games, your biggest problem with libraries will either be that **gcc** can't find a static library or perhaps the library doesn't exist on your system. When playing games from binary, your library woes will be either be that **ld.so** can't find the library or the library doesn't exist on your system. So it makes some sense to talk about how **gcc** and **ld.so** go about finding libraries in the first place.

gcc looks for libraries in the ``standard system directories'' plus any directories you specify with the *−L* option. You can find what these standard system directories are with **gcc −print−search−dirs**

ld.so looks to a binary hash contained in a file named `/etc/ld.so.cache` for a list of directories that contain available dynamic libraries. Since it contains binary data, you cannot modify this file directly. However, the file is generated from a text file `/etc/ld.so.conf` which you can edit. This file contains a list of directories that you want **ld.so** to search for dynamic libraries. If you want to start putting dynamic libraries in `/home/joecool/privatelibs`, you'd add this directory to `/etc/ld.so.conf`. Your change doesn't actually make it into `/etc/ld.so.cache` until you run **ldconfig**; once it's run, **ld.so** will begin to look for libraries in your private directory.

Also, even if you just add extra libraries to your system, you must update `ld.so.cache` to reflect the presence of the new libraries.

5.3.4. Finding Out What Libraries a Game Depends On

Most commercial Linux games will be dynamically linked against various LGPL libraries, such as OpenAL or SDL. For these examples, Bioware's NeverWinter Nights <http://nwn.bioware.com> will be used.

To find out what libraries a game uses, we can use the "`ldd`" command. Cd to `/usr/games/nwn`, or wherever you installed it and take a look at the files. You should see a file called `nwmain`; this is the actual game binary. Type "*ldd nwmain*" and you'll see:

```
$ ldd nwmain
    linux-gate.so.1 =>  (0xffffe000)
    libm.so.6 => /lib/libm.so.6 (0x40027000)
    libpthread.so.0 => /lib/libpthread.so.0 (0x40049000)
    libGL.so.1 => /usr/lib/libGL.so.1 (0x4009b000)
    libGLU.so.1 => /usr/X11R6/lib/libGLU.so.1 (0x40103000)
    libmss.so.6 => not found
    libSDL-1.2.so.0 => /usr/lib/libSDL-1.2.so.0 (0x40178000)
    libc.so.6 => /lib/libc.so.6 (0x401ff000)
    /lib/ld-linux.so.2 (0x40000000)
    libGLcore.so.1 => /usr/lib/libGLcore.so.1 (0x40319000)
    libnvidia-tls.so.1 => /usr/lib/libnvidia-tls.so.1 (0x409f1000)
    libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0x409f3000)
    libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0x40a01000)
    libdl.so.2 => /lib/libdl.so.2 (0x40acd000)
    libstdc++.so.5 => /usr/lib/libstdc++.so.5 (0x40ad1000)
    libgcc_s.so.1 => /usr/lib/libgcc_s.so.1 (0x40b88000)
    libasound.so.2 => /usr/lib/./libasound.so.2 (0x40b90000)
```

ldd shows all the libraries a dynamic executable relies on, and shows you where they are. It also "pulls in" the dependencies of the dependencies. For instance, while NWN does not itself depend on `libnvidia-tls.so`, the Nvidia supplied libGL on my system does.

Missing libraries?

In the example above, we can see that `nwmain` wants `libmss.so.6`, and the linker cannot find it. Usually, a missing library is a crash waiting to happen. There is one other thing to consider though: The majority of games are actually launched by a "wrapper", a shell script that performs some magic prior to launching the game. In the case of NWN, the wrapper is called `nwn`. Let's take a look at that now:

```
$ less nwn
#!/bin/sh

# This script runs Neverwinter Nights from the current directory

export SDL_MOUSE_RELATIVE=0
export SDL_VIDEO_X11_DGAMOUSE=0

# If you do not wish to use the SDL library included in the package, remove
# ./lib from LD_LIBRARY_PATH
export LD_LIBRARY_PATH=./lib:./miles:$LD_LIBRARY_PATH

./nwmain $@
```

This script sets up some environment variables, then launches the game binary with whatever command line options we added. The relevant part here is the environment variable called "LD_LIBRARY_PATH". This is a way of adding to the linkers search path. Try copying the line to your shell and seeing what happens when you re−run ldd.

```
$ export LD_LIBRARY_PATH=./lib:./miles:$LD_LIBRARY_PATH
$ ldd nwmain
    linux-gate.so.1 =>  (0xffffe000)
    libm.so.6 => /lib/libm.so.6 (0x40027000)
    libpthread.so.0 => /lib/libpthread.so.0 (0x40049000)
    libGL.so.1 => /usr/lib/libGL.so.1 (0x4009b000)
    libGLU.so.1 => /usr/X11R6/lib/libGLU.so.1 (0x40103000)
    libmss.so.6 => ./miles/libmss.so.6 (0x40178000)
    libSDL-1.2.so.0 => ./lib/libSDL-1.2.so.0 (0x401ec000)
    libc.so.6 => /lib/libc.so.6 (0x4025e000)
    /lib/ld-linux.so.2 (0x40000000)
    libGLcore.so.1 => /usr/lib/libGLcore.so.1 (0x40378000)
    libnvidia-tls.so.1 => /usr/lib/libnvidia-tls.so.1 (0x40a50000)
    libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0x40a52000)
    libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0x40a60000)
    libdl.so.2 => /lib/libdl.so.2 (0x40b2c000)
    libstdc++.so.5 => /usr/lib/libstdc++.so.5 (0x40b30000)
    libgcc_s.so.1 => /usr/lib/libgcc_s.so.1 (0x40be7000)
```

As you can see, this gives us slighly different results. The NWN library directories have been prepended to the search path, so now the linker can find `libmss.so.6` in the "`./miles`" directory, and also finds the local copy of libSDL first, no longer using the system copy.

There's another benefit of these scripts: they are easily edited to allow you to provide your own copy of a library. Any game−supplied copy of a library such as OpenAL or SDL is likely to be compiled for the lowest common denominator, probably i486 or i686. If you have a Pentium4 or an AthlonXP, you could compile you own version specifically for your processor. The compiler will try to optimise the resulting binary, giving some increase in performance. See the homepage for GCC for more information this at the GCC site.

Making NWN use your system copy is easy. It says so in the wrapper script! Remove "./lib:" from the *LD_LIBRARY_PATH* line, and you're good to go.

Another nice little trick is for games that use OpenAL for their sound output (e.g. Unreal based games: UT, Postal, Rune, etc.). Since the Open Sound System's (OSS) deprecation in favour of ALSA, all Linux distributions I've seen now ship with ALSA support as default, with OSS support actually being supplied via ALSA's compatability modules. The copies of `openal.so` distributed with games often do NOT support ALSA, so making the game use a copy compiled yourself will allow you to use ALSA natively.

---

6. When Bad Things Happen To Good People

Of course we can't cover every Bad Thing that happens, but I'll outline some items of common sense.

There are two types of bad things: random and repeatable. It's very difficult to diagnose or fix random problems that you don't have any control over when they happen or not. However, if the problem is repeatable "it happens when I press the left arrow key twice", then you're in business.

6.1. RTFM!

Read the friendly manual. The `manual' can take on a few forms. For open source games there's the readme files that come with the game. Commercial games will have a printed manual and maybe some readme files on the CD the game came on. Don't forget to browse the CD your game came on for helpful tips and advice.

Don't forget the game's website. The game's author has probably seen people with your exact same problem many times over and might put information specific to that game on the website. A prime example of this is Loki Software's online FAQs located at http://faqs.lokigames.com.

6.2. Look For Updates and Patches

If you're playing an open source game that you compiled, make sure you have the newest version by checking the game's website. If your game came from a distro make sure there's not an update rpm/deb for the game.

Commercial game companies like Loki release patches for their games. Often a game will have MANY patches (Myth2) and some games are unplayable without them (Heretic2). Check the game's website for patches whether you have a problem running the game or not; there may be an update for a security problem that you may not even be aware of.

By the way, Loki now has a utility that searches for Loki Software on your hard drive and automatically updates them. Check out http://updates.lokigames.com.

6.3. Newsgroups

If you don't know what netnews (Usenet) is, then this is definitely worth 30 minutes of your time to learn about. Install a newsreader. I prefer console tools more, so I use tin, but slrn is also popular. Netscape has a nice graphical "point and click" newsreader too.

For instance, I can browse Loki Software's news server with **tin –g news.lokigames.com**. You can also specify which news server to use using the *$NNTP* environment variable or with the file `/etc/nntpserver`.

6.4. Google Group Search

Every post made to Usenet gets archived at Google's database at http://groups.google.com. This archive used to be at http://www.deja.com, but was bought by Google. Many people still know the archive as "deja".

It's almost certain that whatever problem you have with Linux, gaming related or not, has already been asked about and answered on Usenet. Not once, not twice, but many times over. If you don't understand the first response you see (or if it doesn't work), then try one of the other many replies. If the page is not in a language you can understand, there are many translation sites which will convert the text into whatever language you like, including http://www.freetranslation.com and http://translation.lycos.com. My web browser of choice, Opera (available at http://www.opera.com) allows you to use the right mouse button to select a portion of text and left click the selection to translate it into another language. Very useful when a Google group search yields a page in German which looks useful and my wife (who reads German well) isn't around.

The Google group search has a basic and advanced search page. Don't bother with the simple search. The advanced search is at http://groups.google.com/advanced_group_search.

It's easy to use. For example, if my problem was that Quake III crashed everytime Lucy jumps, I would enter "linux quake3 crash lucy jumps" in the "Find messages with all of the words" textbox.

There are fields for which newsgroup you want to narrow your search to. Take the time to read and understand what each field means. I promise you. You won't be disappointed with this service. Use it, and you'll be a much happier person. Do note that they don't archive private newsgroups, like Loki Software's news server. However, so many people use Usenet, it almost doesn't matter.

6.5. Debugging: call traces and core files

This is generally not something you'll do for commercial games. For open source games, you can help the author by giving a corefile or stack trace. Very quickly, a core file (aka core dump) is a file that holds the "state" of the program at the moment it crashes. It holds valuable clues for the programmer to the nature of the crash –– what caused it and what the program was doing when it happened. If you want to learn more about core files, I have a great gdb tutorial at http://www.dirac.org/linux.

At the *very* least, the author will be interested in the call stack when the game crashed. Here is how you can get the call stack at barf–time:

Sometimes distros set up their OS so that core files (which are mainly useful to programmers) aren't generated. The first step is to make your system allow unlimited coresizes:

```
ulimit -c unlimited
```

You will now have to recompile the program and pass the –g option to gcc (explaining this is beyond the scope of this document). Now, run the game and do whatever you did to crash the program and dump a core again. Run the debugger with the core file as the 2nd argument:

```
$ gdb CoolGameExecutable core
```

And at the (gdb) prompt, type "backtrace". You'll see something like:

```
#0 printf (format=0x80484a4 "z is %d.\n") at printf.c:30
#1 0x8048431 in display (z=5) at try1.c:11
#2 0x8048406 in main () at try1.c:6
```

It may be quite long, but use your mouse to cut and paste this information into a file. Email the author and tell him:

1. The game's name
2. Any error message that appears on the screen when the game crashes.
3. What causes the crash and whether it's a repeatable crash or not.
4. The call stack

If you have good bandwidth, ask the author if he would like the core file that his program dumped. If he says yes, then send it. Remember to ask first, because core files can get very, very big.

6.6. Saved Games

If your game allows for saved games, then sending the author a copy of the saved game is useful because it helps the tech reproduce whatever is going wrong. For commercial games, this option is more fruitful than sending a core file or call stack since commercial games can't be recompiled to include debugging information. You should definitely ask before sending a save game file because they tend to be long, but gaming companies usually have lots of bandwidth. Mike Phillips (formerly of Loki Software) mentioned that sending in saved games to Loki is definitely a good thing.

Needless to say, this only applies if your game crashes reproducably at a certain point. If the game segfaults every time you run it, or is incredibly slow, a saved game file won't be of much help.

6.7. What to do when a file or library isn't being found (better living through strace)

Sometimes you'll see error messages that indicate a file wasn't found. The file could be a library:

```
% ./exult
./exult: error while loading shared library: libSDL-1.2.so.0: cannot load shared object
file: No such file or directory
```

or it could be some kind of data file, like a `wad` or `map` file:

```
% qf-client-sdl
IP address 192.168.0.2:27001 UDP Initialize Error: W_LoadWadFile: couldn't load gfx.wad
```

Suppose `gfx.wad` is already on my system, but couldn't be found because it isn't in the right directory. Then where IS the right directory? Wouldn't it be helpful to know where these programs looked for the missing files?

This is where strace shines. strace tells you what system calls are being made, with what arguments, and what their return values are. In my `Kernel Module Programming Guide' (due to be released to LDP soon), I outline everything you may want to know about strace. But here's a brief outline using the canonical example of what strace looks like. Give the command:

```
   strace -o ./LS_LOG /bin/ls
```

The −o option sends strace's output to a file; here, LS_LOG. The last argument to strace is the program we're inspecting, here, "ls". Look at the contents of LS_LOG. Pretty impressive, eh? Here is a typical line:

```
open(".", O_RDONLY|O_NONBLOCK|0x18000)  = 4
```

We used the `open()` system call to open "." with various arguments, and the return value of the call is 4. What does this have to do with files not being found?

Suppose I want to watch the StateOfMind demo because I can't ever seem to get enough of it. One day I try to run it and something bad happens:

```
% ./mind.i86_linux.glibc2.1
Loading & massaging...
Error:Can't open data file 'mind.dat'.
```

Let's use strace to find out where the program was looking for the data file.

```
strace ./mind.i86_linux.glibc2.1 2> ./StateOfMind_LOG
```

Pulling out vim and searching for all occurrences of `mind.dat`, I find the following lines:

```
open("/usr/share/mind.dat",O_RDONLY) = -1 ENOENT (No such file)
write(2, "Error:", 6Error:)   = 6
write(2, "Can\'t open data file \'mind.dat\'."..., ) = 33
```

It was looking for `mind.dat` in only one directory. Clearly, `mind.dat` isn't in `/usr/share`. Now we can try to locate `mind.dat` and move it into `/usr/share`, or better, create a symbolic link.

This method works for libraries too. Suppose the library `libmp3.so.2` is in `/usr/local/include` but your new game "Kill−Metallica" can't find it. You can use strace to determine where Kill−Metallica was looking for the library and make a symlink from `/usr/local/include/libmp3.so.2` to wherever Kill−Metallica was looking for the library file.

strace is a very powerful utility. When diagnosing why things aren't being found, it's your best ally, and is even faster than looking at source code. As a last note, you can't look up information in source code of commercial games from Lokisoft or Tribsoft. But you can still use strace with them!

6.8. Hosed consoles

Sometimes a game will exit abnormally and your console will get `hosed'. There are a few definitions of a hosed console. The text characters could look like gibberish. Your normally nice black screen could look like a quasi−graphics screen. When you press **ENTER**, a newline doesn't get echo'ed to the screen. Sometimes, certain keys of the keyboard won't respond. Logging out and back in don't always work, but there are a few things that might:

- If you don't see any character on the screen as you type in, your terminal settings may be wrong. Try "stty echo". This should let input characters echo again.

* At the prompt, type "reset". This should clear up many problems, including consoles hosed by an SVGAlib or ncurses based game.
* Try running the game again and normally. Once I had to kill Quake III in a hurry, so I performed a Ctl−Alt−Backspace. The console was hosed with a quasi−graphics screen. Running Quake III and quitting normally fixed the problem.
* The commands deallocvt and openvt will work for most of the other problems you'll have. **deallocvt N** kills terminal *N* entirely, so that *Alt−FN* doesn't even work anymore. **openvt −c N** starts it back up.
* If certain keys on your keyboard don't work, be creative. If you want to reboot but the `o' key doesn't work, try using halt. One method I've come up with is typing a command at the prompt and using characters on the screen with mouse cut/paste. For example, you can type "ps ax", and you're sure to have an `h', `a', `l' and a `t' somewhere on the screen. You can use the mouse to cut and paste the word "halt".
* The most regrettable option is a reboot. If you can, an orderly shutdown is preferable; use "halt" or "shutdown". If you can't, ssh in from a another machine. That sometimes works when your console is very badly hosed. In the worst case scenario, hit the reset or power switch.

Note that if you use a journalling filesystem like ext3, reiserfs or xfs, hitting the power switch isn't all that bad. You're still supposed to shutdown in an orderly manner, but the filesystem integrity will be maintained. You won't normally see an fsck for the partitions that use the journalling filesystem.

6.9. Locked System

When a computer "locks", also called "hung", the keyboard and mouse become completely unresponsive. This is a direct consequence of a bug in the Linux kernel. While Linux is known for stability, these things do happen, especiallly for gaming which entails highly synchronized hardware events which occur very fast, even to a computer. When a computer locks, it can be a "hard lock", meaning the kernel has completely stopped functioning. This often indicates misbehaving or faulty hardware. There's no recovery from this kind of lock other than pressing the reset or power button. The lock can also be a "soft lock", meaning that the kernel is still functioning in some capacity. It's possible to recover from this gracefully.

* The first thing you should try is to hit *control−alt−backspace* which kills X. If you gain control of your system, the kernel wasn't really locked in the first place. If this didn't work after a few seconds, you'll definitely want to reboot the system using the following instructions.
* Use *control−alt−delete* to reboot the system. You'll know this worked if you hear the computer beep after a few seconds (this is BIOS saying "I'm OK" during a power on cycle).
* Log into another system and ssh into the hung system. If you can ssh in, reboot or halt the system.
* If you can't ssh into the system, you'll need to use the "magic SysRq key" which is documented in `/usr/src/linux/Documentation/sysrq.txt`. Here's a summary for the x86 architecture (see the documentation for other architectures). Note if your keyboard doesn't have a **SysRq** key, use the **PrintScreen** key:

    1. Hit *alt−SysRq−s*. This will attempt to sync your mounted filesystems so that changes to files get flushed to disk. You may hear disk activity. If you're looking at a console, the system should print the devices which were flushed.
    2. A few seconds later, hit *alt−SysRq−u*. This will attempt to remount all your mounted filesystems as read−only). You should hear disk activity. If you're looking at a console, the system will print the devices which were remounted.
    3. A few seconds later, use *alt−SysRq−b* to reboot the system.
    4. You can hit *alt−SysRq−h* for a very terse help screen.

To use the magic SysRq key, your kernel needs to have been compiled with magic SysRq support. You'll find this option under "*Kernel Hacking | Kernel Debugging | Magic SysRq key*" in whatever kernel config menu you like to use. If the magic SysRq key sequence doesn't shut your system down gracefully, your kernel has crashed hard and you'll need to use the reset or power button to recover.

7. Video Cards

7.1. History

Once upon a time, a company in San Jose, California named 3dfx Interactive was king of the gaming video card market. In October 1996 they released the Voodoo I, which was a phenomenal success. It was the first hardware accelerated card, but only rendered 3D; it had to be piggybacked with a 2D video card. The idea was that 2D rendering was handled by a high quality 2D video card (Matrox was immensely popular at the time) but 3D information (see Glide2, Section 3.1) would be passed to the Voodoo I and rendered, using the Voodoo's fast hardware to perform the necessary graphics calculations. They released the Voodoo Rush in April 1996. It should've been a more powerful card, with a 50MHz GPU and 8MB of RAM. Even better, it was their first combined 2D/3D card, meaning that it freed up a valuable PCI slot (most PC's only had a couple of PCI slots back then) but the Rush wasn't as popular. 3dfx removed the multi−texturing unit from the Rush, and it was outperformed by the Voodoo I. At the time, ATI had their Rage series and nVidia had their Riva 128, but the Voodoo I blew them all away.

This was a good time for Linux. id Software's open sourced the Doom codebase and ported Quake I to Linux (December 1996). We were getting our first tastes of real commercial gaming. Life was simple: you purchased a Voodoo. And it felt good, because 3dfx open sourced their drivers. The king of video cards worked with Linux developers. Not only did we have the best video cards, but the drivers were all open source.

In March 1998, 3dfx released their Voodoo II, with its 3.6GB/sec memory bandwith, 12MB of video memory and 90MHz core. It supported resolutions up to 1024x768. This was 3dfx in its heyday. Like the Voodoo I, the Voodoo II was a 3D only card, and piggy backed with a 2D video card. The Voodoo Banshee was released in September 1998 as a combined 2D/3D card, like the Rush. Despite the faster 100MHz core, the Banshee was outperformed by the Voodoo II because its multi−texturing unit was removed, like with the Rush. And again like the Rush, it wasn't popular. But 3dfx reigned supreme, and nobody could touch them.

In April 1999, the Voodoo III was released. There were a number of Voodoo III's, ranging from a 143MHz core speed to 183MHz. There were TV−out versions. There were PCI and AGP versions (it was the first AGP video card). It was another success, but 3dfx began to lose ground to nVidia, which released their TNT 2. The TNT 2 outperformed the Voodoo II, and accelerated 3D graphics at full 32 bit color, while the Voodoo's were stuck at 16 bit color. But life was still good for Linux. We had a card that was almost neck−to−neck with nVidia, our drivers were open source, and in December 1999, id Software gave us a huge gift: they open sourced the Quake I codebase.

Then nVidia released the GeForce 256 in October 1999. 3dfx's Voodoo IV, its direct competitor, was about a year late which is very bad when you're competing for a bleeding edge market. While nVidia was putting real R&D into their cards, 3dfx was simply adding more and faster RAM. The Voodoo IV and V rendered in full 32bpp color, had great AA support (Section 7.4.3), featured a 2nd GPU, more memory, and was arguably the king of of video cards. However, 3dfx's late release of the Voodoo IV and V coupled with the fact that the GeForce could be had for half the price meant that 3dfx was sinking fast. For Linux, the newest Voodoo's could only accelerate at 16 and 24 bit color. Worse still, the Voodoo V's 2nd GPU was unused by the Linux driver (and to this day, the Voodoo V is functionally equivalent to the single GPU Voodoo IV on Linux). Most Windows users were switching to nVidia, and despite the fact that the nVidia drivers were proprietary, even Linux users began to jump onto the nVidia bandwagon. VA Linux, the largest Linux server vendor, put nVidia into their machines.

Then in April 2000, 3dfx was attacked on a different front: ATI started releasing their first generation Radeons. Until this point, ATI had always been an innovative (they developed their own 3D acceleration chips in 1996, about the same time as 3dfx), but sleepy graphics chipset manufacturer. The Radeons were their first 3D accelerated card that gamers took any real serious interest in. Their Radeons trounced both nVidia and 3dfx. They worked with Linux developers, open sourced all their drivers and were hailed as the great hope for Linux gaming. nVidia came back with fists swinging, and this was all too much for 3dfx. Between losing the benchmark wars to the GeForce and Radeon, their lateness with new cards and high prices, 3dfx lost its market share and didn't have the funds to stay into business. On April 18 2001, they sold most of their assets and technology to nVidia, and in October 2002, they finally declared bankruptcy.

The demise of 3dfx was quite sudden and a slap in the face to the open source community. I still remember my friend Gabe Rosa emailing me with just "*Look at /.*" and seeing the news. It was the 2nd worst day for Linux gaming (the 1st being the demise of Loki). And it was also a shame. 3dfx was getting ready to release a new Voodoo V featuring 4 GPU's which would've trounced the ATI and nVidia offerings, as well as a new card code named "Rampage" which reportedly would've put them firmly back as the king of the hill. There are reports that the Rampage's technology (which was sold to nVidia) went into the GeForce 5900. Not too shabby for 3 year old technology!

At first, things were still simple. Linux gamers would either keep their open source Voodoos, get an open source Radeon or a closed source GeForce. However, with bigger and better games on the horizon, it was only a matter of time before the Voodoos would no longer be a viable graphics card for modern gaming. People were still using Voodoo's, but they were essentially out of the game at this point.

ATI started to release a tremendous number of versions of each video card, and keeping up with them and their terminology started to become very difficult. ATI, together with nVidia, played king of hill. Their products have been neck to neck ever since, with GeForce taking the lead a bit more times than the Radeon. But the Radeon's drivers were open source, so many Linux users stuck by them. Then things got even more complicated.

ATI started becoming more and more reluctant to open source drivers for their new releases, and suddenly, it wasn't clear who the "good guy" was anymore. nVidia's party line was they license some of their GL code from another company, and is thus non−releasable. Presumably, ATI doesn't want to release drivers to keep their trade secrets, well, a secret. And it gets worse. The ATI Linux drivers have been plagued by extremely poor performance. Even when an ATI offering is better than the current GeForce offering for Windows, the card is always trounced by GeForce on Linux. Because of the ATI Linux driver woes, Linux users cannot use MS Windows based benchmarks or card stats. They simply don't apply to us. And that's pretty much where we are right now.

As a last note, the only systematic Linux video card benchmarking effort I'm aware of was done, unfortunately, in March 2001, between a Radeon 32 DDR and a GeForce 2. You can read it for yourself at http://www.linuxhardware.org/features/01/03/19/0357219.shtml, but conclusion is that the GeForce 2 firmly and soundly trounced the Radeon 32 DDR.

7.2. Current Status (1 March 2004)

nVidia's latest offering is the GeForce 5900, based on the NV35 chipset. It's well supported by Linux with high quality but proprietary drivers. nVidia uses a convenient combined driver architecture; their driver will support the TNT 2 all the way up to the GeForce 5900. Although their drivers are closed source, as a company, nVidia has been supportive and good to Linux users.

ATI's has worked with Linux developers for their Radeons up to and including the Radeon 9200, which have 2D and 3D support in XFree86. I'm not entirely sure of the quality of these open source drivers, however, Soldier of Fortune I and Heavy Metal still have opaque texture problems under first generation Radeons. Beyond the 9200, you need to use ATI's binary only proprietary drivers, available in rpm format from ATI's website. It's claimed that these drivers are piss poor; a friend of mine claims his GeForce 4400 outperforms his Radeon 9700 pro. That's shameful.

On paper, and in the Windows benchmarks, the Radeon 9800 trounces the ill−conceived GeForce 5800 and slightly edges out the GeForce 5900. On paper, it's simply the more impressive card. But again, the driver issue makes this information unusable for us. If you have your heart set to buy the best card for Linux, you'll want to go with the GeForce 5900.

7.2.1. SVGAlib Support

As of June 2002, SVGAlib support Radeon cards is shaky. Developers have reported that SVGAlib works on the Radeon 7500, Radeon QD (64MB DDR model) but has problems on the Radeon VE.

I have no information about SVGAlib and the GeForce cards.

7.3. Which Video Card Should I Buy? (1 March 2004)

The answer was very difficult last year, but here's my take on it these days:

1. All GeForce cards require a proprietary driver which will "taint" your kernel. However, all ATI cards beyond the Radeon 9200 also require a proprietary driver that will "taint" your kernel as well.
2. nVidia has proven that they care enough about Linux to write and maintain current and very high quality drivers for Linux. Even when ATI open sourced its video card driver, they played the "we'll make Linux developers write our drivers for us" game. Their current proprietary drivers are below par.
3. The current Radeon 9800 barely beats out the GeForce 5900 in benchmarks and card specs, but Linux users won't benefit from this because of driver issues..
4. ATI has a very long history of dropping support for hardware as fast as they can get away with it.
5. On MS Windows, when the GeForce beat out its main competing Radeon, the review claimed that the Radeon generally had better visuals. I have no idea how this translates to Linux.

Don't get the GeForce 5800. Card reviews claim that it has some serious heat, noise, and dust issues. It's informally called the "dust buster" because of noise its fan makes.

If you absolutely only want open source drivers on your system, the Radeon 9200 is the best card you can buy.

If you have a Linux/Windows dual boot, consider either the Radeon 9800 or the GeForce 5900. The Radeon will be slightly stronger on Windows. The GeForce will be stronger on Linux.

If you have a Linux only system, the GeForce 5900 is your best bet. As of today, the 256MB version comes in at a whopping $350, however, the 128MB version is more reasonable.

7.4. Definitions: Video Card and 3D Terminology

We'll cover video card and 3D graphics terminology. This material isn't crucial to actually getting a game working, but may help in deciding what hardware and software options are best for you.

7.4.1. Textures

A rendered scene is basically made up of polygons and lines. A texture is a 2D image (usually a bitmap) covering the polygons of a 3D world. Think of it as a coat of paint for the polygons.

7.4.2. T&L: Transform and Lighting

The T&L is the process of translating all the 3D world information (position, distance, and light sources) into the 2D image that is actually displayed on screen.

7.4.3. AA: Anti Aliasing

Anti aliasing is the smoothing of jagged edges along a rendered curve or polygon. Pixels are rectangular objects, so drawing an angled line or curve with them results in a 'stair step' effect, also called the 'jaggies'. This is when pixels make, what should be a smooth curve or line, jagged. AA uses CPU intensive filtering to smooth out these jagged edges. This improves a game's visuals, but can also dramatically degrade performance.

AA is used in a number of situations. For instance, when you magnify a picture, you'll notice lines that were once smooth become jagged (try it with The Gimp). Font rendering is another big application for AA.

AA can be done either by the application itself (as with The Gimp or the XFree86 font system) or by hardware, if your video card supports it. Since AA is CPU intensive, it's more desirable to perform it in hardware, but if we're talking about semi−static applications, like The Gimp, this really isn't an issue. For dynamic situations, like games, doing AA in hardware can be crucial.

7.4.4. FSAA: Full Screen Anti−Aliasing

FSAA usually involves drawing a magnified version of the entire screen in a separate framebuffer, performing AA on the entire image and rescaling it back to the normal resolution. As you can imagine, this is extremely CPU intensive. You will never see non hardware accelerated FSAA.

7.4.5. Mip Mapping

Mip mapping is a technique where several scaled copies of the same texture are stored in the video card memory to represent the texture at different distances. When the texture is far away a smaller version of the texture (mip map) is used. When the texture is near, a bigger one is used. Mip mapping can be used regardless of filtering method (Section 7.4.6). Mip mapping reduces memory bandwidth requirements since the images are in hardware, but it also offers better quality in the rendered image.

7.4.6. Texture Filtering

Texture filtering is the fundamental feature required to present sweet 3D graphics. It's used for a number of purposes, like making adjacent textures blend smoothly and making textures viewed from an angle (think of looking at a billboard from an extreme angle) look realistic. There are several common texture filtering techniques including point−sampling, bilinear, trilinear and anisotropic filtering.

When I talk about 'performance hits', keep in mind that the performance hit depends on what resolution you're running at. For instance, at a low resolution you may get only a very slight hit by using trilinear filtering instead of bilinear filtering. But at high resolutions, the performance hit may be enormous. Also, I'm not aware of any card that uses anisotropic texture filtering. TNT drivers claim they do, but I've read that these drivers still use trilinear filtering when actually rendering an image to the screen.

7.4.6.1. Point Sampling Texture Filtering

Point sampling is rare these days, but if you run a game with 'software rendering' (which you'd need to do if you run a 3D accelerated game without a 3D accelerated board) you're likely to see it used.

7.4.6.2. Bilinear Texture Filtering

Bilinear filtering is a computationally cheap but low quality texture filtering. It approximates the gaps between textures by sampling the color of the four closest (above, below, left and right) texels. All modern 3D accelerated video cards can do bilinear filtering in hardware with no performance hit.

7.4.6.3. Trilinear Texture Filtering

Trilinear filtering is a high quality bilinear filter which uses the four closest pixels in the second most suitable mip map to produce smoother transitions between mip map levels. Trilinear filtering samples eight pixels and interpolates them before rendering. Trilinear filtering always uses mip mapping. Trilinear filtering eliminates the banding effect that appears between adjacent mip map levels. Most modern 3D accelerated video cards can do trilinear filtering in hardware with no performance hit.

7.4.6.4. Anisotropic Texture Filtering

Anisotropic filtering is the best but most CPU intensive of the three common texture filtering methods. Trilinear filtering is capable of producing fine visuals, but it only samples from a square area which in some cases is not the ideal method. Anisotropic (meaning 'from any direction') samples from more than 8 pixels. The number of sampled pixels and which sampled pixels it uses depends on the viewing angle of the surface relative to your screen. It shines when viewing alphanumeric characters at an angle.

7.4.7. Z Buffering

A Z buffer is a portion of RAM which represents the distance between the viewer (you) and each pixel of an object. Many modern 3D accelerated cards have a Z buffer in their video RAM, which speeds things up considerably, but Z buffering can also be done by the application's rendering engine. However, this sort of thing clearly should be done in hardware wherever possible.

Every object has a stacking order, like a deck of cards. When objects are rendered into a 2D frame buffer, the rendering engine removes hidden surfaces by using the Z buffer. There are two approaches to this. Dumb engines draw far objects first and close objects last, obscuring objects below them in the Z buffer. Smart engines calculate what portions of objects will be obscured by objects above them and simply not render the portions that you won't see anyhow. For complicated textures this is a huge savings in processor work.

8. Sound

8.1. Which sound card is best?

By the word "best" I mean best for gaming. Gamers want high quality sound for our games with the least amount of tinkering. On the other hand, a musician would have a very different concept of what "best sound card" would mean. If you're a musician, you might want to check out the Linux Audio Quality HOWTO.

Now that Linux is beginning to mature, this question isn't as important as it used to be. Once upon a time, soundcards without onboard MIDI chips (most PCI sound cards) didn't do MIDI. This was mostly a problem for things like xdoom or lxdoom using musserv. These days we have MIDI emulators like Timidity and libraries like SDL which don't require hardware MIDI support. Frankly, I've had many cards and I can't tell the difference between any of them for gaming. If you want to do things like convert a record LP to digital format, then your choice of a soundcard with a professional grade A/D converter is absolutely crucial. For this HOWTO, we'll assume that you're more of a gamer than a studio recording engineer.

Your decision should be based on what will be the easiest to configure. If you already have a card and it works well, that's good enough. If you're in the market to buy a sound card, get something that will take you a second to configure. PCI cards are much easier to deal with than ISA since you don't need to tell their drivers about which system resources (IRQ, DMA, I/O addresses) to use. Some ISA cards ARE plug−n−play, like the Creative AWE−64, and the Linux kernel has come a long way in auto configuring them.

My personal recommendation is any card which has the es1370 or es1371 chip, which uses the es1370 and es1371 sound drivers on Linux. These cards include the older Ensoniq es1370 and newer Creative PCI−128. These cards are extremely cheap and trivial to get working under Linux.

I used to be a fan of the Creative Soundblaster AWE 32, AWE 64 and AWE 64 gold soundcards. These ISA PnP cards are well supported by both OSS and Alsa. They all use the same E−mu 8000 synthesis chip which enables them to play 32 voices simultaneously (they have 32 "channels"). A few notes: First, the Soundblaster AWE HOWTO is very out of date. Second, the AWE 64 and AWE 64 gold can play 64 voices simultaneously, but this is done in software. Creative never released a Linux driver for these cards (and they never released programming information to Linux developers), so Linux users cannot use the extra 32 channels on the AWE 64 and AWE 64 gold. As far Linux users are concerned, all three cards are completely identical (although the AWE 64 gold has gold plated connectors, which are better for sound quality than the more common steel connectors).

The Creative Soundblaster Live! is an extremely popular PCI sound card these days. I've never owned one, so I cannot comment here. However, there have been numerous reports about serious problems with the Live! and AMD motherboards that use the 686b southbridge. A google search should turn up alot of information on this problem.

A more relevent issue is speakers, but even here the difference isn't huge. I've had expensive Altec Lansing speakers perform only slightly better than el−cheapo speakers. You get what you pay for with speakers, but don't expect a huge difference. You'll want to get something with a separate sub−woofer; this does make a difference at a cost of extra power and connector wires.

8.2. Why isn't my sound working?

First of all, it's probably not the game, it's probably your setup. AFAIK, there are 3 options to getting a sound card configured under Linux: the free OSS sound drivers that come with the Linux kernel, the Alsa drivers and the commercial OSS sound drivers. Personally, I prefer the free OSS drivers, but many people swear by Alsa. The commercial OSS drivers are good when you're having trouble getting your sound card to work by free methods. Don't discount them; they're very cheap (like 10 or 20 bucks), support bleeding edge sound cards and take a lot of guesswork out of the configuring process.

There are 5 things that can go wrong with your sound system:

1. Shared interrupt
2. Misconfigured driver
3. Something's already accessing the sound card
4. You're using the wrong driver
5. A permissions problem

8.2.1. Shared interrupt

The first thing to do is to figure out if you have an IRQ conflict. ISA cards can't share interrupts. PCI cards can share interrupts, but certain types of high bandwidth cards simply don't like to share, including network and sound cards. To find out whether you have a conflict, do a **`cat /proc/interrupts`**. Output on my system is:

```
$ cat /proc/interrupts
           CPU0       CPU1
  0:   24185341          0          XT-PIC  timer
  1:     224714          0          XT-PIC  keyboard
  2:          0          0          XT-PIC  cascade
  5:    2478476          0          XT-PIC  soundblaster
  5:     325924          0          XT-PIC  eth0
 11:     131326          0          XT-PIC  aic7xxx
 12:    2457456          0          XT-PIC  PS/2 Mouse
 14:     556955          0          XT-PIC  ide0
NMI:          0          0
LOC:   24186046   24186026
ERR:       1353
```

The second column is there because I have 2 CPU's in this machine; if you have one CPU (called UP, or uniprocessor), you'll have only 1 CPU column. The numbers on the left are the assigned IRQ's and the strings to the right indicate what device was assigned that IRQ. You can see I have an IRQ conflict between the soundcard (soundblaster) and the network card (eth0). They both share IRQ 5. Actually, I cooked this example up because I wanted to show you what an IRQ conflict looks like. But if I did have this conflict, neither my network nor my sound would work well (or at all!).

If my sound card is PCI, the preferred way of fixing this would be to simply move one of the cards to a different slot and hope the BIOS sorts things out. A more advanced way of fixing this would be to go into BIOS and assign IRQ's to specific slots. Modern BIOS'es can do this.

8.2.2. Misconfigured driver

Sometimes, a card is hardwired to use a certain IRQ. You'll see this on ISA cards only. Alternatively, some ISA cards can be set to use a specific IRQ using jumpers on the card itself. With these types of cards, you need to pass the correct IRQ and memory access, "I/O port", to the driver.

This is a sound card specific issue, and beyond the scope of this HOWTO.

8.2.3. Something is already accessing your sound card

Perhaps an application is already accessing your soundcard. For example, maybe you have an MP3 player that's paused? If something is already accessing your card, other applications won't be able to. Even though it was written to share the card between applications, I've found that esd (the enlightenment sound daemon) sometimes doesn't work correctly. The best tool to use here is lsof, which shows which processes are accessing a file. Your sound card is represented by `/dev/dsp`. Right now, I'm listening to an MP3 (not a Metallica MP3, of course...) with mp3blaster.

```
# lsof /dev/dsp
COMMAND    PID USER   FD   TYPE DEVICE SIZE   NODE NAME
mp3blaste 1108    p    6w   CHR   14,3      662302 /dev/dsp
```

fuser is similar; but it lets you send a signal to any process accessing the device file.

```
# fuser -vk /dev/dsp

                     USER        PID ACCESS COMMAND
/dev/dsp             root       1225 f....  mp3blaster
                     root       1282 f....  mp3blaster
```

After issuing this command, mp3blaster was killed with SIGKILL. See the man pages for lsof and fuser; they're very useful. Oh, you'll want to run them as root since you'll be asking for information from processes that may be owned by root.

8.2.4. You're using the wrong driver (or no driver)

There are only two ways to configure your card:

1. Support must be compiled directly into the kernel
2. You must have the correct driver loaded into memory

You can find out which driver your sound card is using by doing "lsmod" or looking at the output of "dmesg". Since sound is crucial for me, I always compile sound into my kernels. If you don't have a driver loaded, you need to figure out what's been compiled into your kernel. That's not so straight forward. Your best bet is to compile your kernel. BTW, let me say that compiling your own kernel is the first step towards proficiency with Linux. It's painful the first time you do it, but once you do it correctly, it becomes very easy down the right, especially if you keep all your old .config files and make use of things like "make oldconfig". See the Kernel HOWTO for details.

If you haven't compiled the kernel yourself, there is an overwhelmingly good chance that your system is set up to load sound drivers as modules. That's the way distros do things. Have everything under the sun compiled as a module and try to load them all. So if you don't see your sound card's driver with lsmod, your card probably isn't configured yet.

8.2.5. Permissions Problem

If the sound card works when you're root but not any other user, you probably have a permissions problem. If this is the case, as root, look at the group owner of the sound card using **`ls −l /dev/dsp`**; it'll probably be *audio*. Then, as root, add your non−root user to the audio group in `/etc/group`. For example, I added the users p and wellspring to group audio on my system:

```
audio:x:29:p,wellspring
```

Don't forget to use **grpconv** if you use shadow passwords (which should be the case on most recent distributions) in order to maintain a consistent group configuration. Then log out and log back in as the non−root user. Your sound card should work. Thanks to James Barton for reminding me to add this to the howto.

9. Miscellaneous Problems

9.1. Hardware Acceleration Problems

XFree86 4.x provides a more centralized and self−contained approach to video. Much of the funkyness like kernel modules for non−root access of video boards is, thankfully, gone.

9.1.1. Hardware acceleration isn't working at all

If you're getting like 1 fps, then your system isn't using hardware 3D acceleration. There's one of two things that can be going on.

1. Your 3D system is misconfigured (more likely)
2. Game X is misconfigured (less likely)

The first step is to figure out which one is happening.

1. If you have X 4.0 (X 3.* users procede to step 2), look at the output of **`X −probeonly`**. You'll see:
   ```
   (II) XXXXXX: direct rendering enabled
   ```
   or
   ```
   (II) XXXXXX: direct rendering disabled
   ```
   where XXXXXXX depends on which video card you have. If direct rendering is disabled, then your X configuration is definitely faulty. Your game is not at fault. You need to figure out why DRI is disabled. The most important tool for you to use at this point is the `DRI Users Guide'. It is an excellently written document that gives you step by step information on how to get DRI set up correctly on your machine. A copy is kept at http://www.xfree86.org/4.0/DRI.html.

   Note that if you pass this test, your system is CAPABLE of direct rendering. Your libraries can still be wrong. So procede to step 2.
2. There is a program called glxgears which comes with the "mesademos" package. You can get mesademos with Debian (**apt−get install mesademos**) or you can hunt for the rpm on http://www.rpmfind.net. You can also download and compile the source yourself from the mesa homepage.

   Running glxgears will show some gears turning. The xterm from which you run glxgears will display "X frames in Y seconds = X/Y FPS". You can compare your system to the list of benchmarks below.
   ```
      CPU TYPE     VIDEO CARD     X VERSION    AVERAGE FPS
   ```
   Compiling Mesa and DRI modules yourself can increase your FPS by 15 FPS; quite a performance boost! So if your number is, say, about 20 FPS slower than a comparable machine, chances are that glxgears is falling back on software rendering. In other words, your graphics card isn't 3D accelerating graphics.

   More important than FPS is having a constant FPS for small and large windows. If hardware acceleration is working, the FPS for glxgears should be nearly independent of window size. If it's not, then you're not getting hardware acceleration.

9.2. Hardware acceleration works only for the root user

9.2.1. XFree86 4.x

If the following lines aren't present in your XF86Config−4 file, put them in:

```
Section "DRI"
    Mode 0666
EndSection
```

This allows all non−root users to use DRI. For the paranoid, it's possible to restrict DRI to only a few non−root users. See the DRI User Guide.

9.2.2. XFree86 3.x

9.2.2.1. Voodoo cards

Voodoo card hardware acceleration only takes place ONLY at 16bpp color and fails silently when starting X in another color depth.

Also, Voodoo cards need the `3dfx.o` kernel module and a `/dev/3dfx` device file (major 107, minor 0) for non−root hardware acceleration. Neither the module nor the device file are used under XFree86 4.x.

10. Emulation and Virtual Machines

Linux gets ragged on a lot because we don't have the wealth of games that other platforms have. Frankly, there's enough games for me, although it would be really nice to have some of the bleeding edge games and classics like Half−life and Carmageddon. Fortunately, we have more emulators than you can shake a stick at. Although playing an emulated game is sometimes not quite as fun as playing it on the native machine, and getting some of the emulators to work well can be a difficult task, they're here, and there's alot of them!

10.1. What is a virtual machine?

A "real computer" provides an operating system many things, including a CPU, I/O channels, memory, a BIOS to provide low level access to motherboard and I/O resources, etc. When an operating system wants to write to a hard drive, it communicates through a device driver that interfaces directly with the hardware device memory.

However, it's possible to give a program all the hardware resources it needs. When it wants to access a hard drive, give it some memory to write to. When it wants to set an IRQ, give it some bogus instructions that lets it think it set an IRQ. If you do this correctly, then in principle, there's no way for the poor application to know whether it's really accessing hardware or tricked by being given resources which simulate hardware. A virtual machine is the environment which tricks applications into believing they're running on a real computer. It provides all the services that a real computer would provide.

VM's were used initially in the 1960's to emulate time shared operating systems, but these days we use them to run software which was written for foreign operating systems, or more commonly, an entire operating system. Because of the nature of the VM, the foreign OS can't tell the difference between operating in a VM or in a "real" machine.

10.2. Apple 8−bit

All the 8−bit Apple][emulators require a copy of the original ROM, for whichever system you want to emulate, in a file. If you search hard enough, you can find file copies of the ROMs for the Apple][,][+,][e,][c and //gs. They are still copyrighted by Apple, and you can only use them legally if you actually own one of these computers.

10.2.1. KEGS

KEGS is an Apple II emulator written by Kent Dickey `<kentd(at)cup(dot)hp(dot)com>` which was originally written for HP−UX, but improved and customized for Linux. It runs under X at any color depth, and supports changeable memory sizes, joysticks, and sound. KEGS boots all Apple II variants, and supports all of the Apple]['s graphics modes. I can't find a working homepage for this application.

10.2.2. apple2 and xapple2

The SVGAlib based `apple2` and X based `xapple2` can emulate any Apple][variant except for the //gs. The interface is a bit funky, but usable. Configuration is also a bit funky; this emulator would benefit from an SVGA or X based configuration tool. It supports the undocumented portion of the 6502 instruction set which some games rely on. `apple2` is currently being maintained by Michael Deutschmann `<michael(at)talamasca(dot)ocis(dot)net>` and seems to be developed at a slow but constant pace. I don't think this application has a homepage.

10.3. DOS

10.3.1. dosemu

dosemu <http://www.dosemu.org> is the canonical DOS emulator on Linux. When you think of DOS, don't think of things like PROCOM PLUS OR OTHER PROGRA~1 WHICH HAVE SHORT NAMES AND ARE IN ALL CAPS. There are some real classics that were written for DOS like Carmageddon, Redneck Rampage and Tomb Raider. dosemu can run these. Unfortunately, it can take alot of effort to get dosemu to work, and of Jan 2002, the sound code is somewhat broken. Not a big deal when you're trying to run Wordperfect or an old database application. It's an absolute show stopper for gaming. Getting dosemu to work well is not easy, but unfortunately, for DOS games it's the best avenue. Good luck. If you have success using dosemu, I would like to hear from you.

10.4. Win16

10.4.1. Wabi

Wabi is a commercial Win16 emulator. That is, it'll run Windows 16−bit applications from a Windows 3.1, Windows 3.11 or Windows for Workgroups 3.11 environment. Wabi was originally created by SCO Unix a long time ago and then was purchased by Caldera sometime in mid year 2001.

Wabi is fast and does a good job for what it does, although I've heard it said that wabi for Solaris is more stable than Linux. It might be useful for playing older Win16 games, but there are three problems:

- You must have a licensed copy of Windows 3.1/3.11 or WfW 3.11.
- Wabi is awfully expensive for what it does.
- Wabi doesn't work under 32bpp or 24bpp color.

Wabi does NOT do DOS itself, but it looks like it can use a DOS emulator as a backend for running DOS programs. There was talk about Wabi 3.0 which would've done Win32 emulation, but AFAIK, this project was shelved indefinitely. I think Wabi will run under Linux on all architectures (can someone verify this?)

10.5. Win32

10.5.1. wine

Wine <http://www.winehq.com>, which bears the GNUish acronym "Wine Is Not An Emulator" is a non−commercial implementation of the Win32 API. The reason why it's not an emulator is subtle and not of much interest to most non computer scientists, so we'll call it an emulator here (it really does run−time translation of calls to the Win32 API to POSIX/X11 calls). Wine has come a long way, and is capable of emulating many important programs, which is great news for Linux users who want this sort of stuff.

Wine does *not* provide the DOS API, so you can't use it to run DOS applications. For that, you should look at dosemu (Section 10.3.1). Wine has never been too good at implementing DirectX, although a number of games are known to work under wine. For gaming you might want to look at winex (Section 10.5.3).

In addition to run−time translation of the Win32 API to POSIX/X11 (it runs Windows applications on Linux), wine also does compile−time tranlation of the Win32 API to POSIX/X11 (it compiles Windows application source code on Linux). In this sense, wine is a Windows−to−Linux porting utility. The x86 architecture isn't required, but is recommended since it allows actual x86 binary execution as well as direct DLL usage).

You can use wine `with Windows', which means that wine uses libraries that actually come with Microsoft Windows itself. This is legal only if you own a copy of Windows which isn't currently being used on a computer. It's said that wine has the best success when run with Windows. You can also run wine without Windows. The people at winehq are writing their own set of libraries called *libwine* which implements the Win32 API with no Microsoft code at all.

Wine was originally licenced under the MIT/X11 license, so it could be used for both commercial and non−commercial purposes. In mid 2002, parts of wine were re−licensed under the LGPL so that it could only be used for non−commercial puposes. This presents a problem for companies like Transgaming (Section 10.5.3) and prompted a fork of wine called ReWind (Section 10.5.2).

10.5.2. rewind

Rewind <http://rewind.sourceforge.net/> was started by Eric Pouech (a wine developer) and Ove Kåven (a winex developer) in response to wine's license change). It started out life as a snapshot of the last version of wine which was completely licensed under the MIT/X11 license. The aim is to keep rewind MIT/X11 based so that companies like Transgaming can offer wine based products.

10.5.3. winex

Winex is released by a company called Transgaming <http://www.transgaming.com>. The developers take wine (see Section 10.5.1) and add DirectX / DirectDraw support. Although winex is commercial, they have an interesting business model.

The end user (you) can download the source code for free. However, for 5 US dollars per month, you can become a subscriber of Transgaming. Being a subscriber of Transgaming gives three major benefits:

- Subscribers can download convenient packaged versions of winex in deb, rpm or tar.gz format whenever they want, including updates. They have also more functionality than the publicly available tarball: the latter is an older version which lacks some of the newest features, like support for copy protected programs.
- There are monthly polls where subscribed users can take votes on what they want winex developers to work on. For instance, they can vote for things like "Improve support for copy protected programs", "Better Installshield support" or "Improve DirectX 8.0 support". As far as I can see, the developers really do listen to the subscriber polls.
- The Transgaming website has a few user support forums. On one hand, they use the most godawful, horrible, confusing, wasteful, moronic format I've ever seen and I hope to god I never see a forum with a format as bad as Transgaming's. On the other hand, you can ask for help and the developers are VERY good about getting around to your answer; their vigilance is quite impressive. Non−subscribers can browse the forums, but only subscribers can post (and therefore, ask for support).

The developers of winex were going to release their Installshield, DirectX and DirectDraw enhancements to wine "every so often". In return, as wine maturation improved, the winex developers were going to take the new versions of wine and use them for winex. However, since the birth of Transgaming, parts of wine have been re−licensed under the more restrictive GNU LGPL license (Section 10.5.1). This basically means that versions of wine that are released past the date of the re−licensing can no longer be used by winex. Therefore, winex will now be based on rewind (Section 10.5.2).

10.5.4. Win4Lin

Win4Lin <http://www.netraverse.com> is a commercial product by Netraverse. Like vmware (Section 10.5.5) it uses the virtual machine approach to running Windows applications, so you'll get a big window from which you can boot Windows and run all kinds of Windows applications. Unlike vmware, Win4Lin only does Windows 95/98/ME, but this turns out to be better for gamers. Because Win4Lin concentrates on these operating systems, reports say that it's faster and does a better job at running games under these operating system than vmware. It's also much cheaper than vmware. The most recent version of Win4Lin as of June 2003 is 5.0. It suffers nevertheless from some limitations:

- It does not support DirectX or DirectDraw, while vmware has "limited" support for DirectX.
- It only supports serial and parallel devices. This is important for people who use USB joysticks. Note that vmware supports up to 2 USB devices.
- As of June 2003, expect to pay $89.99 without printed docs and $99.99 with printed docs. In addition, there isn't an evaluation copy available, although you get a 30 day money back guarantee. However, since it's commercial you do get tech support. vmware is considerably more expensive.
- Like vmware, you're required to have a licensed copy of Win95 or Win98. Win4Lin cannot use an existing Windows installation the way wine can.
- It only runs on x86 architectures.

10.5.5. VMWare

VMWare is a virtual machine that runs multiple operating systems simultaneously on a standard PC: supported OSes include Microsoft ones, Linux, Novell Netware and FreeBSD. You can among others use it to run a MS Windows OS and launch your favourite game there. You can even run another Linux under Linux; useful is you want to test another distro for instance. Amazing! Now for the bad sides. You should definitely have a good configuration in order to run it; they claim the minimum is a 500MHz x86 CPU with 128MB RAM, but a faster processor and at least 256MB RAM seem to be the bare minimum if you want reasonable performance. Not all Linux distributions are supported: newest RedHat's, Mandrake's and Suse's are, but you're out of luck if you have an other version and/or distribution (like Debian). Moreover, vmware has only limited support for DirectX, and you might not be able to play recent games.

See http://www.vmware.com for more information. It's not very cheap (about 300$ for the Workstation version), but you can get a 30 day evaluation copy.

10.5.6. What should I choose?

First of all, you should try an emulator. Although some games may work with wine, you'll probably get the most success with winex: its DirectX support is constantly improving. As of version 3.1, the DirectX 8 support is nearly complete, but this may not be the case with older DirectX versions (are consequently older games).

You might also try a virtual machine like Win4Lin or VMWare instead of an emulator. If your goal is to run Win95/98/ME applications on Linux, without USB and on the x86 architecture, Win4Lin's cost and focus on Win95 type OS's make it a better choice than vmware. However, if you must have USB support or run Linux on a platform other than x86, vmware is your only option.

Now if your goal is to run Win95 type OS games under Linux, Win4Lin almost seems better than vmware. The show−stopper is the fact that vmware has limited DirectX support while Win4Lin has none. This fact alone makes both Win4Lin and vmware unsuitable for most hardcore gaming purposes. But if you're going to give it a try, you're more likely to have success with vmware.

11. Interpreters

11.1. SCUMM Engine (LucasArts)

Lucasarts wrote an engine for point and click adventures named SCUMM (Script Creation Utility for Maniac Mansion). They wrote many graphical adventures using SCUMM, like their famous Monkey Island series (all three). Ludvig Strigeus `<strigeus(at)users(dot)sourceforge(dot)net>` was able to reverse engineer the SCUMM format and write an interpreter for SCUMM based games that compiles under Linux and Win32 named scummvm <http://scummvm.sourceforge.net/>. Their website is very good, and chock full of any kind of information about SCUMM and playing these games under scummvm.

A compatibility page is maintained at the scummvm website. FWIW, I've been able to finish many of the games that are listed as 90% done with no problems. scummvm is rock solid, and allows you to purchase SCUMM based Lucas Arts games, copy the data files to your hard drive and play them under Linux. As of February 2002, I've been following their cvs, and this project is undergoing constant development. Kudos to the scummvm team.

11.2. AGI: Adventure Gaming Interface (Sierra)

The older Sierra DOS graphical adventure games used a scripting language named AGI (Adventure Gaming Interface). Some examples of games written in AGI would be Leisure Suit Larry I (EGA), Space Quest I and II, King's Quest II, Mixed−Up Mother Goose and others. These games can be played using sarienon> <http://sarien.sourceforge.net>, an open source interpreter for AGI games.

Sarien was written in SDL, so it should run on any platform that can compile SDL programs. In addition, there are versions for DOS, Strong−Arm based pda's, QNS (holy cow! embedded gaming!), MIPS based systems and SH3/4 based Pocket PC's. The developers are clearly out of their minds (in a good way!). Sarien also has numerous enhancements not found in the original games, like a Quake style pull−down console, picture and dictionary viewer, enhanced sound and support for AGDS, a Russian AGI clone. Sarien is under development and the developers have been very good about documenting the Sarien internals if anyone wants to get involved in hacking it.

11.3. SCI: SCript Interpreter or Sierra Creative Interpreter (Sierra)

The newer Sierra graphical adventure games (we're talking about the late 80's here) used an interpreter named SCI. There were many versions of SCI since Sierra was constantly improving its engine. The original SCI games were DOS based, but Sierra eventually started releasing Win32 SCI based games. Some examples of games written with SCI are Leisure Suit Larry 1 (VGA), Leisure Suit Larry 2−7, Space Quest 3−6, King's Quest 4−6, Quest For Glory 1−4 and many others. Compared with AGI games, SCI adventures have better music support, a more complex engine and loads of bells and whistles.

Many SCI based games (games written in SCI0) can be played using freesci, available at http://freesci.linuxgames.com. Like Sarien, FreeSCI has many graphics targets including SDL, xlib and GGI, so this program can compile and run under an incredible number of platforms. The developers have done a fantastic job of documenting and FAQing their application.

11.4. Infocom Adventures (Infocom, Activision)

The Z−machine is a well documented <http://www.gnelson.demon.co.uk/zspec/index.html> virtual machine designed by Infocom to run their interactive fiction games. This allowed them to write game data files in a cross platform manner, since only the engine itself, the Z−machine, would be platform dependent. Z−machine went through a number of revisions during the lifetime of Infocom, and two further revisions (V7 and V8 created by Graham Nelson) after the Infocom's demise. The later versions even supported limited sound and graphics!

One of the most popular Z−machine interpreters is Frotz <http://www.cs.csubak.edu/~dgriffi/proj/frotz/>. This excellently done page has many nice links for interactive fiction fans. Frotz is GPL, runs all versions of Z−machine and will compile on most versions of Unix. Frotz has spawned many forks, like a version for PalmOS and Linux based PDA's.

jzip <http://jzip.sourceforge.net/> is another very popular Z−machine interpreter that will run V1−V5 and V8 Z−machine data files. jzip is very portable; it compiles on all Unices, OS/2, Atari ST and DOS.

There are actually many other Z−machine interpreters like nitfol and rezrov (written in Perl!). Each interpreter has its own set of strengths, and you can find links to them on the home pages for Frotz and jzip.

11.5. Scott Adams Adventures (Adventure International)

Scott Adams is, arguably, the father of interactive fiction. Although he himself was inspired by the first piece of interactive fiction, Adventure, Scott brought adventuring to the masses. His games were available for Atari, Apple 2, Commodore, Sorcerer, TI, and CPM. His company, Adventure International, released a number of much loved games between 1978 and 1984 before folding. He recently released a new game (a Linux version is not available) but since the decline of adventuring, he has pretty much kept out of the gaming industry.

Alan Cox wrote scottfree, a Scott Adams adventure game file interpreter for Unix. Using scottfree and any of the Scott Adams data files which can be downloaded from Scott's website <http://www.msadams.com/> you can enjoy these classics.

11.6. Ultima Underworld: The Stygian Abyss (Origin, Blue Sky Productions)

The Underworld Adventures project <http://uwadv.sourceforge.net/> is an effort to port the 1992 classic, Ultima Underworld: The Stygian Abyss, to modern operating systems like Linux, MacOS X, and Windows. It uses OpenGL for 3D graphics, SDL for platform specific tasks and is released under the GNU GPL. Underworld Adventures provides an impressive graphics system which uses the original game files, so you'll need the original game disk to play.

Underworld Adventures also provides a bunch of tools for you to display the level maps, tools for examining uw1 conversation scripts and more.

11.7. Ultima 7 (Origin, Electronic Arts)

Ultima 7 is actually 2 games: part I (The Black Gate) and part II (Serpent Island) which uses a slightly enhanced version of The Black Gate's engine. In addition, an addon disk was released to both part I (The Forge Of Virtue) and part II (The Silver Seed).

A team of people developed Exult <http://exult.sourceforge.net/> which is an open source interpreter that will run both parts of Ultima 7 and their addon disks. Exult is written in C++ using SDL, so it will compile on any platform that can compile SDL programs. It also features some enhancements over the original versions of the Ultima VII engine. You'll need to purchase a copy of Ultima 7 to play. The developers have no plans on extending Exult to interpret the other Ultimas since the engines changed so radically between releases.

The Exult team has also been hard at work creating a map editor, Exult Studio, and a script compiler that will let users create their own RPG in the Ultima style.

11.8. System Shock (Electronic Arts, Origin)

System Shock is a classic first person shooter/adventure from 1994, which puts it as a contemporary of Doom. However, its engine is much more feature rich than the original Doom: for example, System Shock had 3D sprites, free look and a facility to have objects on top of each other, giving the illusion of a full 3D map, like Quake. Game reviewers agree that this game has the features of Quake with a story−line more compelling than Half−life. The System Shock engine was optimized for sophistication, while Doom's engine was optimized for throwing lots of monsters at you; a completely different appoach. Quite impressive for such an old game!

The System Shock Hack Project <http://madeira.physiol.ucl.ac.uk/tsshp/sshock.html> is an attempt to update the game for modern operating systems. The project uses SDL and is released under the modified BSD license. While you need the original game files to play SSHP, it should work with the System Shock demo, which is freely available.

12. Websites And Resources

12.1. Meta gaming websites

These are some resources for Linux gamers no matter what kind of game you enjoy to play.

The Linux Game Tome: http://www.happypenguin.org
: About the games themselves.

Linuxgames
: Linux gaming news

http://www.holarse.net
: Linux meta gaming site for German speaking folk.

Mobygames
: A database of all known computer games. It's very complete and is one of my favorite sites.

12.2. Commercial Linux Game Resources

12.2.1. Where to buy commercial games

ebgames <http://www.ebgames.com> no longer officially sells Linux software. They stopped selling Linux games and distributions at around the same time Loki Software declared bankruptcy, which is a shame because they had the lowest prices on Linux games I've ever seen. However, occasionally, they'll have things like Code Warrior or Redhat Linux on sale.

Tux Games: http://www.tuxgames.com
: Your one stop shop for buying any commercial Linux game (software vendors like Tribsoft and Loki have online shops at their websites too).

12.2.2. Who Used To Release Games For Linux

These are companies that used to release games for Linux but for whatever reasons aren't actively involved in Linux games anymore.

Loki Software: http://www.lokigames.com
: As the company that brought CTP and Quake3 to Linux, Loki was the father of Linux gaming. They were one of the first and had, by far, the most titles (I own ALL of them). Loki ported games to Linux, mostly using the SDL library. Loki's death in January 2002 was the biggest setback Linux has ever had in its attempt to capture the general desktop market. Linuxgames.com has a nice Loki timeline at http://www.linuxgames.com/articles/lokitimeline

Tribsoft: http://www.tribsoft.com
: Tribsoft released Jagged Alliance 2, an excellent rpg/strat which claimed 2+ weeks of my life. There were slated to release Europai Universalis, Majesty and Unfinished Business. However, as of 3Jan01, Mathieu Pinard of Tribsoft said that he was taking a break and Tribsoft would no longer release games for awhile. He'll still support JA2 but don't expect patches or updates.

MP Entertainment: http://www.hopkinsfbi.com
: MP Entertainment released Hopkins FBI, my favorite game ever released for Linux. More violent than Quake. More nudity than Hustler. More camp than Liberace. It's a comic book on your monitor. They were slated to release Hopkins FBI II and a few other titles, but it's been a few years since the

announcements with no sign that the games are coming. They've ignored all my attempts at finding out more information, so I have to conclude that MP Entertainment is in the same status as Tribsoft. You can still purchase or download a demo of Hopkins FBI from their website. If anyone has more information on this company or the author of Hopkins FBI, please contact me.

Phantom EFX: http://www.phantomefx.com
: They offer Reel Deal Slots, which is very nicely done! I'm not much for card/gambling games, but this game is impressive! Because their Linux guy quit the company, Reel Deal Slots is their first, and so far, last release for Linux.

12.3. Other Resources

This section has URL's that should be mentioned but didn't have a separate section within the howto, so I list them here as a kind of appendix.

Linux Game Publishing: http://www.linuxgamepublishing.com
: Linux Publishing doesn't sell directly to the public, but provides professional game publishing to authors of publishing. I think this means disk copying, packaging and selling to retailers.

XFree86 Homesite: http://www.xfree86.org
: XFree86 home page

Linux Game Development Center: http://lgdc.sunsite.dk/index.html
: This is the canonical website for people who want to program games under Linux. It's a clearing house of information that contains well written articles on all aspects of game programming (not necessarily Linux specific), links to important game programming resources, interviews, reviews, polls and lots of other stuff. It's hard to imagine a better website on the subject.

Linux Gamers' FAQ: http://www.icculus.org/lgfaq/
: Despite the astounding fact that the Linux Gamers' FAQ doesn't mention the Linux Gamers' HOWTO as a resource anywhere in their text, I regard the FAQ as a good companion to this HOWTO. I've tried to keep game specific information in this HOWTO at a minimum. The FAQ takes the opposite approach; they mainly focus on the games themselves, including game specific problems and where to get Linux games in the first place. The FAQ and HOWTO are complementary in this regard, and I've tried to not reproduce their content. Despite the authors being a bit surly, their effort with the FAQ is very good. If you want a general source of information on game specific questions, the FAQ is a fantastic place to start with. In addition, the FAQ keeps a fairly large database of Linux Games.

Linux Audio Quality HOWTO: http://www.linuxdj.com/audio/quality/
: This HOWTO is mainly of interest to musicians who want professional or semi professional sound cards for the recording and making of music on a computer. The information is very detailed, and perhaps overkill for gamers.