The Linux Gamers' HOWTO

Peter Jay Salzman

p@dirac.org

2002−02−22 v.0.9.10

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@dirac.org>. My web page is www.dirac.org/p and my Linux pages are at 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

1.2. Acknowledgements

Thanks to Mike Phillips who commented extensively on the howto. Thanks to Dmitry Samoyloff, <dsamoyloff@yandex.ru>, for translating this document into Russian. It blew my mind when he told me that he was translating my words to Russian.

1.3. Latest Version and Translations

The latest version can be found at cvs.sourceforge.net/cgi−bin/viewcvs.cgi/lgh/LG−HOWTO, but this is my own personal working copy. You can get the most recent polished version (whatever that means!) from www.linuxdoc.org and www.dirac.org/linux/writing.

Dmitry Samoyloff, <dsamoyloff@mail.ru>, is the maintainer of the Russian translation of this HOWTO. The most recent version can be found at linuxgames.hut.ru/data/docs/HOWTO/LG−HOWTO−ru.html.

---

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 (poker, solitaire, etc); the type of game you'd play on a table top with friends, but the program simulates 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 simulates 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'. They were self contained executables on disks (even casettes). These days we're a bit more sophisticated than that. Now there's usually a data file and an interpreter. The interpreter reads data files and provides the gaming interface. The data file is the actual game, and is often implemented by a scripting language. So for example, you you could have the two Scott Adams datafiles "The Count.dat" and "Voodoo Castle.dat". To actually play the games, you'd invoke the scottfree interpreter with the name of the datafile you wish to play.

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.

They became popularized by Scott Adams, who is widely considered to be the father of text adventuring. You can play Scott Adams adventures using scottfree, the game file interpreter written by Alan Cox, and the old data files, which are now shareware and can be download from Scott Adams' website.

Text adventures climaxed in the 80's with Infocom. There are many Infocom interpreters available for Linux; the most popular one being frotz. You still need the data files, and these are all still owned and considered commercial property by Activision.

As computer graphics became easier and more powerful, text adventures gave rise to graphic adventures. The death of 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 of them are considerably more complex, and there's often a fine line between sims and strats (see the next section). A good example would be Heavy Gear III or Flight Gear. These days sims and strats are nearly indestinguishable, but a long time ago, sims were all real time while strats were all turn based. This is somewhat 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 or illegal drug selling operation (SimCity or DrugWars). The scenario can also be total 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 Games released the 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, look 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. Some are set in fantasy (Hexen), others are science fiction (Quake II), and some are set in the present day `real world' (Soldier Of Fortune).

Just 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 discard their engines and put them into 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 FAKK.

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.

The canonical RPG on Linux is Rogue (the ncurses library can be traced back to the screen handling routines that were written for Rogue!) and its infinite variants like Zangband and Nethack (which has infinite variants itself). Some of them are quite complicated and are great feats of programming. There seems to be a deficiency of commercial RPGs on Linux. If you don't count all the rogue variants, there also seems to deficiency of open source RPGs as well.

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.

---

3. Libraries

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

3.1. What is Glide2?

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

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 natively using Glide2 (Myth II, Descent III)
- indirectly: uses Mesa built with a Glide2 backend to simulate OpenGL (Rune, Deux Ex, 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.* users to use Voodoo I, II and III cards under Linux.

Since Glide2 accesses the video board directly, applications that use Glide2 will either need to be run by root. A way around this was to create the kernel 3dfx module. This module (and the corresponding device /dev/3dfx) allows Glide2 hardware acceleration for non−root users of non−setuid graphics 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.* (most people use XFree86 4.*). 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?

Glide3 is not a direct API used to program a game with, unlike Glide2. It exists only to support DRI on Voodoo III, IV and V boards under XFree86 4.*. None of the games which use Glide2 will work with Glide3. This shouldn't be a surprise since Glide2 and Glide 3 support different video boards and different versions of XFree86. The only video card that can use both Glide2 (under XFree86 3.*) and Glide3 (under XFree86 4.*) 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.*, 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 is defined and maintained by the Architectural Revision Board (ARB), an organization whose include 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: Tools for system independent window event handling (mouse event handling, keyboard events, etc.).

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). The important difference being that since OpenGL is open (and DirectX is closed), games written in OpenGL are much easier to port to Linux while games written using DirectX at this point in time are impossible to port to Linux.

3.4. What is Mesa?

Mesa <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.*, 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 with X 4.* with DRI (Direct Rendering Infrastructure <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 doesn't have to be sent over a socket to the X server.
- On single processor machines the CPU doesn't have to change context between X 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. (For instance, it relies on root privileges rather than creating the kernel infrastructure necessary for secure access). It provides support for (at this time) a couple cards which are not well−supported in DRI. Particularly, 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.

As a side note, until the G400 stuff is *really* fixed in DRI, it's also the better G400 support but hopefully that will not be an issue by the time this HOWTO is published.

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 arcane and complicated. A program that simply connects to an X server, puts up a window and does nothing else could be a complicated 40 line program with arcane and very long named functions. Because of this, there are lots of libraries which hide the details of xlib programming. Some of these libraries focus on drawing in windows (like SDL). Other libraries are more concerned with widgets within windows (eg pulldown menus, radio buttons and text boxes); these types of libraries are called widget sets. 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 king of the Unix world, but was very expensive to license. The Open Group opened up Motif's license for non−commercial use, but it was too little too late. People now use Lesstif, a Motif clone.

3.9. What is SDL (Simple DirectMedia Layer)?

SDL <www.libsdl.org> is a library by Loki Software's 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. The 40 line arcane program I mentioned in the xlib section can easily be written in

6 lines of straightforward code using SDL.

The most striking part of SDL is that it's a cross platform library. Except for a few details about header files and compiling, a program written in SDL will compile under Linux, MS Windows, BeOS, MacOS, MacOS X, Solaris, IRIX, FreeBSD, QNX and OSF. There are SDL extentions 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's was the lead programmer for Loki Software, which used SDL in all of its games except for Quake3.

3.10. What is GGI?

GGI is seemingly now−defunct project which aimed to implement the graphics abstraction layer in lower level code, putting graphics hardware support in one place, bringing higher stability and portability to graphics applications and replacing SVGAlib, fb, and X servers dealing directly with hardware.

GGI was supposed to have a kernel module inserted into the Linux source code but Linus thought their code wasn't ready for production kernels. There was brief talk about making *BSD their main platform. It's been ages since I've heard anything about GGI. Does anybody have a working URL for this project?

3.11. 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 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 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 would you want to simulate text mode from a graphical environment? This allows us to run graphical things in console, like allowing us to choose what font we want the console to display (which is normally determined by BIOS). Imagine having a console font of Comic Sans MS? There's a good Frame Buffer howto available from LDP.

3.12. What is OpenAL?

OpenAL <www.openal.org> aims to be for sound what OpenGL is for graphics. Jointly developed by Loki Software and Creative Labs, it sets out to be a vendor neutral and cross platform API for audio. It is licensed LGPL and the specs can be had for free from the OpenAL website. OpenAL is fully functional, but now that

Loki Software is no more its future development is questionable.

3.13. What is DirectX?

DirectX is a bunch of multimedia (2d and 3d graphics, sound, animation) API's for Microsoft's Windows OS, first developed by MS in 1995. Direct3D is to MS Windows as SDL is to Linux, except that Direct3D proprietary, closed source, very high level and only meant to compile and be used under Windows under the x86 architecture, whereas SDL is open source, lower level and extremely portable to other platforms. As of February 2002, the most recent version of DirectX is 8.1. The component of DirectX which is responsible for 3D graphics is called Direct3D.

We mention it here because it is a competing technology to open technologies like SDL and OpenGL/OpenAL. Many games have, historically, been unportable to Linux because they have been written in Direct3D. However, recently there have been Direct3D ports to Linux like Loki Software's Heavy Gear II. Also, Direct3D is important to people who play Windows games under Linux via wine, winex or one of the virtual machines like vmware. In particular, the whole point of winex is to provide better support for Direct3D which isn't very developed under plain wine.

A company named realtechVR started an open source project called the "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. The DirectX Port is open source and you can get their latest cvs from their sourceforge page at http://sourceforge.net/projects/dxglwrap>.

4. Definitions: Video Card and 3D Terminology

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

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.

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.

4.3. AA: Anti Aliasing

Anti aliasing is the smoothing of jagged edges along a rendered curve or polygon. Pixels are square 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 are now 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.

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.

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 (see below). Mip mapping reduces memory bandwidth requirements since the images are in hardware, but it also offers better quality in the rendered image.

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.

One thing to keep in mind is that when I talk about `performance hits', 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.

4.7. 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.

4.8. 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.

4.9. 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 these before rendering, twice as much as bi−linear does. 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.

4.10. 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 way. 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.

4.11. 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.

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.

5. 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`).

5.1. Getting information about your X system

5.1.1. 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
```

5.1.2. 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 Modetiming Howto for more information.
- the "dot clock" your videocard is running at.

5.1.3. 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 resolution

: Width and Height

: color bpp

: Depth

and a few other things which are interesting but not immediately relevent to our subject, like "Window Gravity State" which tells where new windows tend to be placed by the window manager.

5.1.4. 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.).

5.1.5. Getting information about your 3D system

glxinfo gives lots of useful information about OpenGL (whether direct rendering is being used or not, the currently installed versions of glx and mesa), vendor/renderer strings, the GL library files being used and more.

6. Various Topics

6.1. Memory Type Register Ranges

Starting with Pentium class processors and including Athlon, K6−2 and other CPUs, there are Memory Type Register Ranges (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.

6.2. 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.
- Don't run a window manager (wm). Some wm's like twm don't take up much CPU cycles, but still rob you of performance. Some window managers like enlightenment will definitely produce a noticeable slow down. To run a game without a wm, you modify .xinitrc in your home directory. Here is what my .xinitrc looks like:

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

This file tells X what client to run upon starting. Usually this is your wm, and/or a desktop manager (GNOME or KDE). Comment out the lines containing a wm and 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. Note that this is for people who use `startx' to start X.

I never use things like gdm or run−level 5 (so I'm not positive here), but I suspect that if you do, you'll need to do things a bit differently. My best guess is to go to single user mode (run−level 1) by:

```
# telinit 1
```

then edit .xinitrc, then go back to run−level 5 by

```
# telinit 5
```

Then when you stop playing, go to run−level 1, modify .xinitrc then go back to run−level 5. I don't use this stuff, so I'm not sure, but you may need to kill gdm. I'd appreciate some feedback on this.

Kill all not−essential processes. Of course you'll have to do this as root. A better way to do this than typing "ps ax", getting ntpd's pid, and sending it a SIGKILL (with kill −9) is to make use of pidof:

```
# kill −9 `pidof ntpd`
```

However, an even better alternative is to use 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 scrip 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 be running nothing which 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.

6.3. 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.

6.3.1. 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.

6.3.2. 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.

6.3.3. 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 presense of the new libraries.

7. 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.

7.1. 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 faqs.lokigames.com.

7.2. 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 updates.lokigames.com.

7.3. 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`.

7.4. Google Group Search

Every post made to Usenet gets archived at Google's database at groups.google.com. This archive used to be at 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 www.freetranslation.com and translation.lycos.com. My web browser of choice, Opera (available at 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 girlfriend (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 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.

7.5. 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 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.

7.6. 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 get long, but a company like Loki Software has 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.

7.7. 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 libraries: 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 Initialized 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 "4" is the return value of the call. 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 occurances 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 of /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!

7.8. 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 won't work, but there are a few things that will:

- 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 gave it the 3 fingered salute. 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 fasion, but the filesystem integrity will be maintained. You won't see an fsck for the partitions that use the journalling filesystem.

---

8. Hardware

I'm no Tom's Hardware or Anandtech, and don't have access to all the wealth of hardware that's out there. Contributions and information to fill out this section would is welcome. This stuff changes very often, and peoples' experience with hardware would be useful.

8.1. Which video card is the best?

If you're using Linux, you must be smart enough to know that there isn't a plain answer to this question. There seem to be 3 choices for hardware accelerated 3D these days:

1. 3dfx: Voodoo cards
2. Nvidia: GeForce
3. ATI: Radeon

According to Tom's Hardware and Anadtech, the Radeon is king when playing at very high resolution (as in 1600x1200), at 32bpp, in Windows. Otherwise the GeForce is king. There are two problems with this. We don't normally play at 1600x1200/32bb, and we don't play on Windows (at least I don't).

There aren't many recent video card benchmarks out for Linux. The most recent one I've seen is from March 2001 at www.linuxhardware.org/features/01/03/19/0357219.shtml. Considering the dearth of benchmarks out there, this needs to be taken as a canonical benchmark, so I simply quote their conclusion:

> At this point the performance numbers tell a pretty simple story, if it's raw speed you are looking for, the GeForce 2 is your choice. There is very little performance drawback to running your favorite games in Linux instead of Windows with this card. It provides a truly impressive performace across the board. The Radeon's performance will almost definitely improve as the DRI drivers mature, but for now, especially for the impatient, it is simply not a good choice for the hard core 3d gamer.
>
> If, however, you are a graphics designer, and want a card with impeccable 2d image quality, with 3d graphics only a secondary priority, the Radeon is your best bet. The DRI drivers, even in their current state, are quite usable. For 2d only users, XFree86 4.0.2 provides production quality 2d drivers. The GeForce thoroughly trounced the Radeon in the Xmark performance test, so if you aren't running at a ultra high resolution, or aren't that picky, the GeForce is once again a better pick.

Now for my own input. The Radeon is a pretty amazing card. It's what I use, and I have yet to see a game that needs more power than the Radeon is able to provide. However, the OpenGL renderer for the Radeon is buggy, although the only games I've seen that suffer greatly are Loki Software's Heavy Metal (which is, regrettably, unplayable) and Soldier Of Fortune. Hopefully the people doing Mesa for the Radeon will fix this very soon since the Radeon is the best option for people who don't want to rely on the closed source, proprietary GeForce.

Now about the Voodoo cards. Unfortunately, 3dfx was bought out by nVidia, so these cards are a dead end market. If you're out to play the bleeding edge games like Rune or Tribes2, you'll want the Voodoo 3, 4 or 5. Preferably the 4 or 5. I think the Voodoo 5 is basically a Voodoo 4 with a second processer. However, this

processor is not utilized by the Linux driver, and rumor says that the Linux 3dfx driver will never support it. So as far as Linux is concerned, the Voodoo 4 and 5 are the same card. All the drivers, Glide2 library and OpenGL renderers for the Voodoo cards were open sourced by 3dfx before they when under. It is an embarrasment to the Linux and open source community in general that this company failed.

8.2. Which sound card is best?

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 quite a big fan of the Creative Soundblaster AWE 32, AWE 64 and AWE 64 gold soundcards. They are ISA, but are plug−n−play. A couple of issues to note. First, the Creative AWE HOWTO is very out of date. Second, AFAIK, Creative never released a Linux driver that uses the AWE 64's extra 32 voices (and they never released programming information for it either). So to a Linux users, the AWE 64 and 32 are nearly identical sound cards. If anyone has more information about the differences that a Linux user would see between the AWE 64 and 32, I'd like to hear from you.

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.

9. Miscellaneous Problems

9.1. Hardware Acceleration Problems

XFree86 4.0 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 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 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 gears which comes with the "mesademos" package. You can get mesademos with Debian (**apt−get install mesademos**) or you can hunt for the rpm on rpmfind.net. You can also download and compile the source yourself from the mesa homepage.

   Running gears will show some gears turning. The xterm from which you run gears will read "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 gears 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 gears 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.*

If the following lines aren't present in your XF86Config 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.*

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.*.

9.3. 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 4 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

9.3.1. 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.

9.3.2. 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. (I should write about how to pass info to the driver).

9.3.3. 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.

9.3.4. 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.

---

10. Emulation and Virtual Machines

Linux gets ragged on alot 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 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. 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.

10.1.1. KEGS

KEGS is an Apple II emulator written by Kent Dickey <kentd@cup.hp.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.

10.1.2. 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, too; 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@talamasca.ocis.net> and seems to be developed at a slow but constant pace. I don't think this application has a homepage.

10.2. DOS

10.2.1. dosemu

dosemu <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.

10.3. Win16

10.3.1. 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.

Reports are that 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?)

10.4. Win32

10.4.1. wine

wine, which bears the GNUish acronym `Wine Is Not An Emulator' is a non−commercial implementation of the Win32 API using Unix calls and X. The reason why it's not an emulator is subtle and not of much interest to non computer scientists, so we'll call it an emulator here. Its homepage is www.winehq.org". Wine has come a long way, and is capable of emulating many complicated and important programs, which is great news for Linux users who want this sort of stuff. Note this doesn't mean that Wine can run pure DOS programs; it can't. It implements the Win32 API. Not DOS. For that, see dosemu.

Wine is both a complete Windows emulator (run−time translation of Win32 calls to POSIX/X11) as well as a Windows−to−UNIX porting kit (compile−time translation of Win32 calls to POSIX/X11). x86 architecture is not required, but is recommended since it can allow actual x86 binary execuation as well as direct DLL support/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 has never been too good at implementing DirectX. However, see the section on winex.

10.4.2. winex

Winex www.transgaming.com seeks to implement DirectX within wine's framework. They have an interesting business model. Their code will, at first, be commercial. They'll release code every so often under the open−source wine license.

10.4.3. Win4Lin

Win4Lin is a commercial product by Netraverse. (www.netraverse.com/products/win4lin30/. It uses the virtual machine approach, so you'll get a big window from which you can boot Windows and run all kinds of Windows applications. I've never used Win4Lin, but since it's a virtual machine, I imagine it does Direct X and games just fine. There are a few problems with using Win4Lin:

- It's not cheap. As of January 2002, expect to pay $80 without printed docs and $90 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.
- You are required to have a licensed copy of Win95 or Win98. Win4Lin cannot use an existing Windows installation the way wine can.
- It can only run on x86 architectures.

10.4.4. VM Ware

---

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@users.sourceforge.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 sarien <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 freesci.linuxgames.com. Like Sarien, FreeSCI has many graphics targets including SDL, xlib and GGI, so this game 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)

11.5. Scott Adams Adventures (Adventure International)

11.6. 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.

12. Websites

12.1. Meta gaming websites

The Linux Game Tome: www.happypenguin.org

About the games themselves.

linuxgames: www.linuxgames.com

Linux gaming news

www.holarse.net

Linux meta gaming site for German speaking folk.

12.2. Commercial Linux Game Websites

12.2.1. Where to buy commercial games

Tux Games: 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).

ebgames: www.ebgames.com

They've got the best prices for Linux games I've ever seen.

12.2.2. Who Used To Release Games For Linux

Loki Software: 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: 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: 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 Liberacce. 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, please contact me.

Phantom EFX: www.phantomefx.com

They offer Reel Deal Slots, which is very nicely done! I'm not much for poker/card/gambling games, but this game is impressive. They have other releases, but have since dropped Linux from their list of OS's to publish games for.

12.3. Other Websites Of Note

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: 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: www.xfree86.org

XFree86 home page

12.3.1. Game Development

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.

12.3.2. Medium Level Libraries

Clanlib <www.clanlib.org

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, threadding 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.