The Unofficial XviD FAQ

Last updated: Tuesday April 6, 2004, 16.00 (gmt+1)

FAQ CONTENTS

Official location="http://www.vslcatena.nl/~ronald/docs/xvidfaq.html"
Maintained by Crusty
If you want to contact me, send me a private message on the Doom9 forum.
Get a (less often updated) PDF version of this FAQ here or here
Tip: Use CTRL+F in your browser for a quick lookup of keywords or options.
Note: Each hyperlink to a third party site will open in a new window.

Part B: Getting it working

Part C: Options, Schmoptions....

Foreword

Welcome to the Unofficial XviD FAQ!
Up till last year, XviD documentation had been given a relatively low priority by the developers. Development tended to focus completely on codec improvement and explanations of XviD options tended to be rather short and cryptic (if there were any at all) and usually one had no other source for explanations than the internet forums. Then in the spring of 2003 a rather lively discussion came about on Doom9's XviD forum about rewriting and creating new documentation and from that moment things went uphill. Guides for specific builds (like Doom9's and Snowbeach's guides) came available and the precise function of many options became the object of fruitful discussion. This FAQ represents the result of many, many hours of sifting through websites, documentation, and public and private discussions with developers and users. We hope you will find it useful.

Part A: Introduction

Back to top

A1. What is XviD?

You had to ask. XviD is an MPEG-4 compliant Video codec. It either encodes a video-file into a (hopefully) MPEG-4 compliant data stream which can be saved into a container-file like an .avi, .ogm, .mp4 or others. By itself it can't do any encoding, it needs another tool to feed the data-to-encode to it. A typical example would be VirtualDub or virtualDubMod (often shortened to VDub and VDubMod). The decoder part usually comes built-in in the encoder, hence the term codec (**CO**der/**DEC**oder) . There are also standalone decoders. These can only playback files, not encode them. The best-known of these is Nic's decoder filter. With it you can play XviD video with any avi-capable media player. Typical examples of these are Windows Media Player(WMP), Media Player Classic(MPC), BsPlayer and Zoomplayer, but there are many more.
XviD is **open source** and is released under the GNU General Public License. Basically this means that anyone can take the source code, use it and alter it any way they see fit to, as long as the result is *also* released under the same license.

A2. What's a codec?

A codec is short for **CO**der/**DEC**oder and is basically no more than a small program. A codec is always a part of a chain of programs that take a *data stream* from a source, do some funky magic on it, and deliver it to some other data stream. This is called input and output by programmers.
It takes a data stream as *input*, either compresses or decompresses it (depending on what you told it to do), and then delivers the altered data stream to it's *output*. The data stream from the input can be from a file, like an uncompressed videofile or a file compressed with another format (like MPEG-1), but also a live video-feed from a

digital camera or a TV. In the first case the input is generated by the file system driver of the operating system, which takes the file sitting on the harddisk and feeds it to the codec (or more accurately to *the program chain* that contains the codec), and in the second case the input is generated by the driver from the hardware device that delivers the life video-feed (like the driver from a TV-card or the video-in from your VGA card).

The chain of programs that a codec is a part of is different depending on what you're doing. If you are encoding a file, a typical chain of programs would be (from input side to output side):
-Avisynth, which takes the input delivered by the file system driver and does some preprocessing to it, like cropping, deinterlacing and resizing (and much more depending on your avisynth script)
-Virtualdub, which accepts the output from avisynth as it's input, converts it to a format that it can understand, and does some more preprocessing on it (that's what virtualdub's filters do). Virtualdub then delivers that data stream to the next part, the codec.
-The codec takes the output from virtualdub as it's input and does the actual compressing to the format you want in the end. It then delivers it's output back to virtualdub.
-Virtualdub then basically hands the newly compressed data over to the file system driver which puts the altered data stream back to the harddisk, in the form you described in virtualdub (which it also delivered to the file system driver).
Get it?

With playback, the principle is really the same, just the programs are a bit different:
-The file system driver delivers the data in the file you want to play as a data stream to the input of the mediaplayer (which could be any player like WMP, MPC, Bsplayer, Zoomplayer etc.)
-The Mediaplayer looks at the datastream, checks what it is, and then takes a look into either it's internal codec list (Media Player Classic does this for instance) or it looks at an external codec list (in the registry of windows for example) to see if it can find the needed codec(s). If it can't find it, it goes belly up with a message like *'unable to locate appropriate decompressor'* or something like that and just sits there waiting for you to install the proper codec.
-If it does find, it feeds the data stream into it.
-Then the codec does it's decompressing stuff and feeds the decompressed data stream back to the player program, which then feeds it to the display driver of your graphics card. The graphics card then puts it on it's output, which depending on what you chose it to be, could be either the vga out to your pc monitor, or the Tv-out of your graphics card (if you happen to have one with a TV-out).

Note 1: Depending on your choices the media player itself can do some altering to the data stream, like increasing the brightness and/or contrast or other stuff you can think of.

Note 2: Usually, a file contains more than a single data stream. A typical file like an avi contains a single video stream and one or more audio streams. Also, a more advanced

container format like Ogm or Matroska (.Mkv) or MP4 (.mp4) can contain **many** data streams, including chapter info, multiple subtitles, and possibly even multi-angle video in the future. These advanced container formats usually need another small program, a *splitter*, to be installed. It's not really a codec but it's needed to make the media player understand how the data streams are stored in the container format. Windows comes with built-in support for avi's and Mpeg's (.mpg and .mpeg), so most media players can play those without any problems. Many other container formats exist, each with their own unique peculiarities.
Examples of these are .ram, .rm, .mov, .ogm, .ogg, .mkv, .mp4, .wma, but the list certainly doesn't end here.

So in short, typical chains are:
Encoding: File system--> Avisynth-->virtualdub-->codec-->virtualdub-->file system
Decoding: File system--> Mediaplayer(checking container format)-->container format filter-->codec-->Mediaplayer(altering brightness/contrast/etc)-->VGA card driver-->Display (Monitor, TV, etc.)

Typically, a codec codes only for one type of data it's specifically designed for. However, some formats are **interchangeable** (also called *interoperable*) because they are built around the same **standard**. Usually files encoded by encoders using the same standard can be played by more than one decoder.
Examples of these standards are MPEG-1, MPEG-2, MPEG-4, AAC, and MPEG Audio Layer 3 (known to everybody as *mp3*). Codecs using these standards are (more or less) interchangeable. between the standards there is usually no interopability whatsoever.
So an MPEG2-codec can usually decode any MPEG-2-stream (like the video from a DVD or a MPEG-2 file downloaded from the internet), but it cannot decode a DivX file because that is MPEG-4. Similarly, an mp3 codec like Lame does not know how to handle ogg vorbis or wma audio.

A3. What is MPEG-4?

MPEG-4 is a video compression standard developed by an industrial group called 'Moving Picture Experts Group' or 'MPEG' for short. MPEG-4's official description by the MPEG is 'a standard for *Very Low Bitrate Audio-Visual Coding*'. Many of the standards developed by this group are accepted standards by the ISO (International Standards Organisation) and are therefore also called ISO-standards. MPEG-4 is ISO/IEC standard #14496. XviD follows MPEG-4 Part 2, also known as MPEG-4 Visual or ISO standard #14496-2. The MPEG standards are widely accepted by the media industry and you can find them in VCD's, DVD's, mp3's and all sorts of other applications.
(the latest addition to the MPEG-4 family of standards for video coding is MPEG-4 Part 10/AVC (Advanced Video Coding) also known as **H.264**. XviD does not support H.264 though.)
Basically, MPEG-4 is a video compression standard with many extensions that are

specifically designed to achieve very high compression on *standard* video content. Standard means in this case real-world video, not stuff like cartoons, pixar films, and anime for instance. (That's part of the reason why these type of footage are so hard to encode properly; MPEG-4 was not designed for it).
MPEG-4 became a proper MPEG standard in 1998, and became an ISO standard in 2001. A technical overview of the MPEG-4 standard can be found here.

Typically MPEG-4 applications are implemented either in software or in hardware. In software usually means in some form of encoder or decoder program, while hardware implementations are typically electronic chips or circuitry designed to encode and/or decode MPEG-4.
Typical MPEG-4 codecs are DivX, 3ivx, Quicktime MPEG-4 and of course XviD. While these are all MPEG-4 *compliant*, that does not necessarily mean they can all play each others files properly. An example of this is the handling of more than 1 B-frame, which XviD can handle but DivX cannot. Another example would be XviD's 3-warppoint GMC.
MPEG-4 implementations in standalone hardware players, like DVD-players that can also play DivX, are usually not real hardware implementations but a sort of semi-hardware implementations. They load software contained in a flashable chip called an Eeprom into a more general-purpose digital processing chip, that then decodes the file it finds on the inserted disc.
The software in the Eeprom is called *firmware* and can sometimes be upgraded to incorporate better support for other MPEG-4 files besides just DivX.

A4. Some MPEG-4 basics

To compress a high quality 2 hour video clip to something that can fit on a single CD obviously takes some serious compression tactics. MPEG-4 does this partly by removing information that we don't notice and partly by *transforming* the raw pixel data into a mathematical approximation of that data. The approximation is close enough for us mere humans not to notice the difference between the source and the result (well, at least in theory).
First the **color space** (the way the pixels, their color and their brightness are stored in each frame) gets converted from whatever it was to a color space called YV12. The human eye is less sensitive in the color (also called chroma) field and much more sensitive in the brightness (also called Luminance or Luma) field, so the brainpeople from MPEG sat around a table and decided that Chroma information is *less* important than Luma information.
The result is that, while a Luminance value is stored for every pixel, a Chroma value is stored for every four pixels (Please note that one Chroma value consists of two separate, one-byte values; see 'YV12' in the glossary).
To illustrate the next step in the process, best is to quote from the Compression FAQ @ www.faqs.org:

"The basic scheme is to predict motion from frame to frame in the temporal direction, and then to use DCT's (discrete cosine

transforms) to organize the redundancy in the spatial directions. The DCT's are done on 8x8 blocks, and the motion prediction is done in the luminance (Y) channel on 16x16 blocks."

The next step in the process is derived from the notion that much of the information in a moving picture can be considered 'static' if you properly take into account the motion of that information. To illustrate this, consider a TV-report of a formula 1 grand prix race. Most of the picture consists of the cars moving about a bit inside the TV-frame as the cameraman is constantly trying to keep the car(s) centered in his view. The picture of the car changes little throughout the shot, (maybe loses focus now and then) but it sort of 'wobbles' around the center of the frame.
In fact, testing showed that most real-life video footage falls in this category. *As long as you capture the motion correctly*, image compression can be *very high and still remain accurate*.
After the motion search a technique called ***D**iscrete **C**osine **T**ransformation* or **DCT** is used to compress texture information. (At decoding time this process is reversed and is called ***I**nverse **D**iscrete **C**osine **T**ransformation* or ***I*DCT**)
The codec divides the picture up into 8x8 blocks on which the DCT process is done. 4 of these 8x8 blocks are grouped together to form a single **macro**block. The information in those blocks is considered *detail* and it can be high or low. DCT takes the detail in those blocks and performs a technique called the *quantization process* on it. The 8x8 blocks then no longer contain pixels but frequency values. High frequencies represent high detail and low frequencies represent low detail. Those frequencies are then recomputed using a **Quantization Matrix** that tells the codec what frequencies are to be dropped and when. This is a rather complex process and requires quite an elaborate explanation. You can find a nice one here.

After the image is converted to detail signals by DCT those signals are divided (coefficient cutting) to make it smaller. The more signal you cut off, the more detail you lose and more artifacts (blocks, ringing aka mosquito noise) you get. The quantizer is the detail removal factor (DivX3/Nandub thus calls quantizers DRF). The higher the quant, the more detail gets cut off, and the lower video quality will be.

Another important concept you have to be aware of is that there are different types of frames in MPEG-4. Typically these are called Keyframes (also called I-frames or INTRA-frames), P-frames (for Predicted frames, also called inter-frames) and B-frames (for BI-directional frames). There are other types of frames but they're not often used (you will read about a few of them further down).

-Keyframes are considered all by themselves and all the picture information in them is encoded. You can always look at a Keyframe without having to know a single bit of the other frames.

-P-frames only contain the *difference* from the previous frame. Say you had a picture of a stoplight in the previous frame and in the current frame it went from red to green, then this frame would only contain the altered information, which is the red turning to black in one part of the picture and black turning to green in a part below that. of

course that's a lot less information to encode so P-frames are normally much smaller than Keyframes. So the P-frame tells the codec that all the other information it needs is contained in a previous frame that has to be taken as a reference. That previous frame can be either a Keyframe or another P-frame.
A typical chain of frames would look like this: IPPPPPIPPPPPIPPPPP, with I being Keyframes.

-B-frames are Bi-directionally predicted frames and they reference not only the previous frame but also the one following it (unless the following frame is a Keyframe). The codec tries to reference the previous frame, the next frame, or a mix between the two, and choses the way that works best. If the frame references only the previous frame it becomes a P-frame, otherwise it becomes a B-frame.

A5. Who develops XviD?

The developers of XviD are just ordinary people, who happen to enjoy writing code for video compression. The developers work on XviD in their spare time, for no compensation other than the fun of doing it and the occasional 'thank you'. If you wonder why XviD isn't perfect, or bugs sometimes appear without immediate fixes, it's because the developers are normal people and cannot commit their lives to the project. They're all volunteers, and you should remember this when asking them questions or requesting features.

A6. Is XviD better than DivX?

Well we like to think so, otherwise we'd be using DivX, right? Read the codec comparison at http://www.Doom9.org/index.html?/codecs-203-1.htm and try to make up your own mind. While XviD comes out on top in this codec comparison, that doesn't necessarily have to mean it's the best solution for your particular need. If you do not want to read the whole comparison, at least read the conclusion, because it can point you to other codecs that might better serve any special need you have.

A7. What should I know before I start with XviD?

If you just want to **play** XviD files, head over to section B1 and grab the latest Xvid decoder. As an alternative you can use any generic MPEG-4 decoder, but most of the ones around don't have full support for all of XviD's advanced features.

For encoding XviD you should know that it's an MPEG-4 compliant codec and what that means. It also helps to know a bit about it's predecessors mpeg-1 and mpeg-2. A good place to start would be here or here, but there are many other good sources. You should also know more about avi-files, which is by far the most used standard for saving encoded video files. A good start would be by reading the Avi overview on this site or the faq here. There are also newer and more flexible file formats available today

like OGM, MP4 and Matroska, and some other formats over the horizon.
Another good place to read-up on related topics would be http://www.dvdrhelp.com/faq for all sorts of concerns, or the DVD FAQ here. You should also have read at least a few of Doom9's excellent guides, especially the Gordian Knot guides, which are great for newbies. Try to encode a few films by using these guides and then come back here to learn a bit more.

A8. Why are there new binaries on a daily basis?

XviD is still in beta phase and under heavy development.

A9. Why is there not one single version of XviD, or a simple sequential version increase like with DivX?

XviD is not made by a single company or single person, but by a group of people scattered around the world that communicate and develop XviD through the internet. Development is constant, but the developers generally only develop the source code. To get a working codec out of this you have to *compile* the source code for the type of computer and Operating System you want to use it on. There is no official authority that 'releases' new builds as a standard. Add to that the fact that XviD has been considered alpha for quite a long time and has only recently went to beta status.
Since there could be many changes to the source code made in a single day it's quite useless to speak of XviD 0.7 or something like that. Only since the developers have started to work to a version 1.0, which is supposedly to be the first really official and stable version, you can expect these type of versions to become more commonplace. Don't be surprised though if development continues for quite some time until it goes 1.0. You could easily end up with a version called 1.0-RC11 or something similar.

A10. I don't know how to compile/what compiling is!

Don't panic. Luckily for us mortals there are some developers out there that **do** release compiled versions of the source code. These compiled versions are called 'builds', or 'binaries'. The most well known of these are compiled by **Koepi, Nic, Umaniac** and more recently **Gamr**.
While some of these builds are tested before they are released by the person who build them, others can be completely untested and be completely broken. Using them is, and has been, at your own risk.

A11. Help! I don't want to kill my computer!

Don't worry. While it can't be ruled out, the chances of a build making your computer going belly-up are very slim. Usually using-at-your-own-risk means that it can screw up the end-result in one way or another or not being MPEG-4 compliant. When a certain feature doesn't work the way it's supposed to, it is usually termed 'breaking' that

feature, for instance 'breaking filesize-prediction' or something like that. Basically this means you can get all sorts of weird results from these builds, ranging from Darth Vader's Helmet turning purple every 5 seconds up to getting a 2 GB file when you wanted it to be just 1 GB.

A12. So what's the difference between these 'tested' and 'untested' builds?

Basically twofold:
-Since testing takes some time, tested versions don't come out every day but are released less often. Untested versions are just versions that compiled, i.e. they could be built without giving an error during compilation.
-The other, and more significant difference is that the tested versions will have been more straightened out than the untested versions. A big 'Oops!' in a tested version will very likely have been noticed by the person testing it before release and is more likely to have been fixed. So, relatively speaking, a 'tested' version can be considered more stable than an 'untested' version. It's not an absolute law though.

A13. Now I'm really confused. Which build should I be using?

The one you feel comfortable working with. That's a silly answer, I know.
But it's basically a question that has only a personal answer.
By far the most widely used builds are made by either Koepi or Nic. While some of these had obvious bugs, later ones in general have been proven quite stable. of course 'quite' is not 'totally', but they're working on it. If you want stability, MPEG-4 compliance and compatibility, wait for 1.0 to be released. of course we don't know how long a wait that could be.

A14. But I want to use XviD now! Can't you recommend a build?

Like I said it's up to personal taste. Some builds have more features, but could also have more bugs. Other, older builds can be more 'proven' but may have less features or are slower. It really depends on how much of a risk you're willing to take. You can go for a more stable version if you're satisfied with it's features, or for a more feature-rich version and take the risks.

A15. What are the risks of using a less stable build exactly?

What you really have to worry about are:
-Visual artifacts when using another, usually newer build of XviD to decode.
-Not playing at all in newer builds.
While not playing at all is relatively uncommon, there are many reports of visual artifacts when switching from one build to another. That doesn't mean it's bound to happen, but there is a chance. It's how much risk you're willing to take that really defines your options.

A16. So any build you can recommend?

You're really asking for my opinion. In fact, if you ask this in any XviD forum, all you'll get are opinions. Look around and see for yourself.
My opinion:
-You *could* go for Koepi's stable version of 04-10-2002. It is considered very stable, but it's over a year old and doesn't have many of the advanced features. I do not recommend it. Use it if you like doing 30 Mph on highways.
Right now XviD has reached 'Release Candidate status' which basically means that the developers are trying to find and fix as many bugs as possible before XviD goes 1.0. Then all previous versions become obsolete and you should undoubtedly switch to 1.0. The latest and recommended build as of 6 april 2004 is Koepi's XviD-1.0RC4 build.

A17. Can I start using XviD now and will I be able to play files created with today's build once the final version comes out?

As XviD is creating MPEG-4-compliant video streams, you can play your movies encoded today with any MPEG-4-compliant decoder (i.e. any future version of XviD, or other MPEG-4 players).
While this means that *downwards compatibility* is more or less assured, occasionally an alpha build *can* introduce a bug that could produce artifacts when played with future versions. *Upwards compatibility*, i.e. older versions playing files encoded with newer versions, is **not** assured.

Part B: Getting it working

Back to top

B1. What do I need to play back XviD encoded videos?

To properly play XviD video you need at least a small program called a decoder installed on your computer (Note: These are part of a more general genre of programs called *filters* in techie-talk).
For windows, the current decoder of choice is:
The 1.0-RC4 package from koepi's site at http://koepi.roeder.goe.net/ (get the whole package and not the RC1 decoder, it's not recent)

As an alternative you can get Milan's FFDShow filter
(As a general rule, get the latest version, even if it is alpha)
You can also get a post-RC1 XviD.ax from sysKin: at his place
(Note that you will also need the XviDcore.dll for playback, sysKin's file is not a complete standalone decoder)

Although FFDShow has more options, it is not based on the XviD project and may contain incompatibilities from time to time. It is advised to try both and see what works best for you.
(These filters do not decode audio! only video! XviD does not decode or deal with audio in anyway. For audio look here.)

B2. What are the minimum hardware requirements for XviD?

The minimum requirement for encoding is a computer with a 32-bit CPU. However, specific builds can have many more requirements. Most builds have some optimizations for SSE(2), 3DNow and other CPU structures, so newer CPU=better performance.
The codec itself hardly takes up any space at all, and if you need to free up space on your hard drive to install, you probably need a bigger harddrive. ;^)
The minimum requirements for decoding (playing) an XviD file are not completely known.
I've seen a Pentium II-500 with 256 MB Ram play XviD but it was stuttering now and then.
And you have to realize that certain encoding options, especially Qpel but also GMC and B-frames, can increase the requiered CPU power for proper playback. The resolution is also a major factor: a 320x240 requires less CPU power than a 1280x720 (encoded resolution, not the resolution you play it back). To be on the safe side, I'd say you need a PC with at least a 600+ MHz CPU and at least 64 MB ram on Windows 98, and 128 on Windows 2000/XP.
Note 1: The XviD codec supports post-processing which will improve the visual quality at the cost of CPU cycles. The faster your computer is the more post-processing options you can turn on. If you find your video plays choppy try disabling all the post-processing options first.

B3. Any hardware recommendations for encoding?

Yes. Get a fast machine. How fast? As fast as your money can buy. Easy eh? Encoding speed depends on raw CPU power, CPU-to-Memory busspeed, Memory speed, and Harddisk speed, in that particular order. You can have a fancy Graphics card that cost you $500 but it won't help you encode one bit faster.
Concerning other parts of your machine, it should be noted that **every** extra component requires at least some CPU power, even if it's not used. Especially sound and network cards (including 'on-board' ones) can depend heavily on the CPU to do much of their processing. And while Creative soundcards tend to have a lower CPU usage, there are many other reasons not to use them. Another thing is historical: XviD tends to be slightly more optimized for AMD CPU's than for Intel CPU's. But that can change any time.

B4. Where can I get XviD builds?

Nic's binaries: http://nic.dnsalias.com/
Koepi's binaries: http://koepi.roeder.goe.net/
uManiacs binaries: http://XviD.hopto.org/
Gamr binaries:http://XviD.gamrdev.com/

Please note that Koepi's and Gamr builds are the most recent (at this time), Nic's and uManiacs binaries are currently **not recommended**.
(Also you should be aware that uManiacs and GamR builds are so-called insta-builds. These are builds that are automatically compiled overnight anytime the CVS-tree (where all the developers put new code) changes and are completely untested. **They can be completely broken because of new bugs and are unsupported**.)

B5. How do I encode using XviD?

Depending on where you want to get the source video from, you may need additional soft- and/or hardware. A typical example would be ripping a DVD, for which you need a movie-DVD (duh!), a DVD-Rom Drive, a DVD-Ripping program like DVDDecrypter and several other tools to convert the data to something Virtualdub can digest.
For starters, I'd recommend **the Gordian Knot rip and system pack**. It has all the tools you need for making your very first own XviD file and it has plenty of additional software for you to fool around with in case you want to learn some more tricks.
You can find it at http://www.Doom9.org/XviD.htm including a guide on how to use it. As of Feb 13th 2003, the interface has changed, however the options described in the guide work the same way.
A more recent version of the guide is available at http://www.Doom9.org/index.html?/XviD-vdub-final.htm. It covers the 1.0beta build, that has many more options and again a changed interface.

B6. I want to compile XviD myself. How do I do that?

Download the source code, unzip it to a directory of your choice, and then go to the '/xvidcore-x.x/doc/' directory. Read the file labelled 'INSTALL' in that directory to get the latest info on how to compile XviD.
Some useful links:
How to compile XviD using M$ Visual C++ 6.0
Discussion on using the Intel compiler
www.bloodshed.net
www.mingw.org
www.cygwin.com
A few caveats when compiling:
-**Don't use any .NET compiler**..they're too buggy right now.

-There's currently no possible way to use 64-bits optimizations. All the speed-hungry parts of XviD are written in assembler. And the source has to be rewritten to make use of 64-bits registers and extensions (mostly cut&paste and register adjusments, but it still has to be done properly and thoroughly). Xvid will compile on x86_64 in pure c, however this removes all speed advantages of running a 64-bit binary, it is slower than running the assembly optimized 32-bit binary. Other stuff like the VfW interface is not done in assembler and you can compile it with 64-bits optimizations, but it won't make any difference really.

B7. How do I get the latest CVS sourcecode?

Download WinCVS here, install it and use
cvs -d:pserver:anonymous@cvs.xvid.org:/xvid login ..to login anonymously.
Then use:
cvs -d:pserver:anonymous@cvs.xvid.org:/xvid co -R -r dev-api-4 xvidcore ..to get the dev-api-4 branch.
Read more about it at www.xvid.org.
You can also check out this page at the Xvid Project homepage for the most up-to-date info.

B8. How do I use XviD under Linux?

The most commonly used applications for encoding with Xvid under GNU/Linux are 'transcode' and 'mencoder' (part of Mplayer) which come with the proper communication module for the codec (be sure to get an up to date one for the latest versions (>beta) from http://ed.gomez.free.fr or from transcode's or mencoder's cvs).
These are command-line applications but have many frontends like Auto-transcode, Kmencoder or gmencoder, providing them with a GUI. Another Linux tool specifically for ripping DVD's is dvd::rip.
Xvid configuration is not handled the same way as in windows, so don't look for the configuration GUI, it's not there. Configuration is usually handled from the encoding application with the help of configuration files. The same options apply of course but whether or not you have access to them depends mostly on the application you use at the moment. Check out this Doom9 forum thread for more on that.
Another transcode link

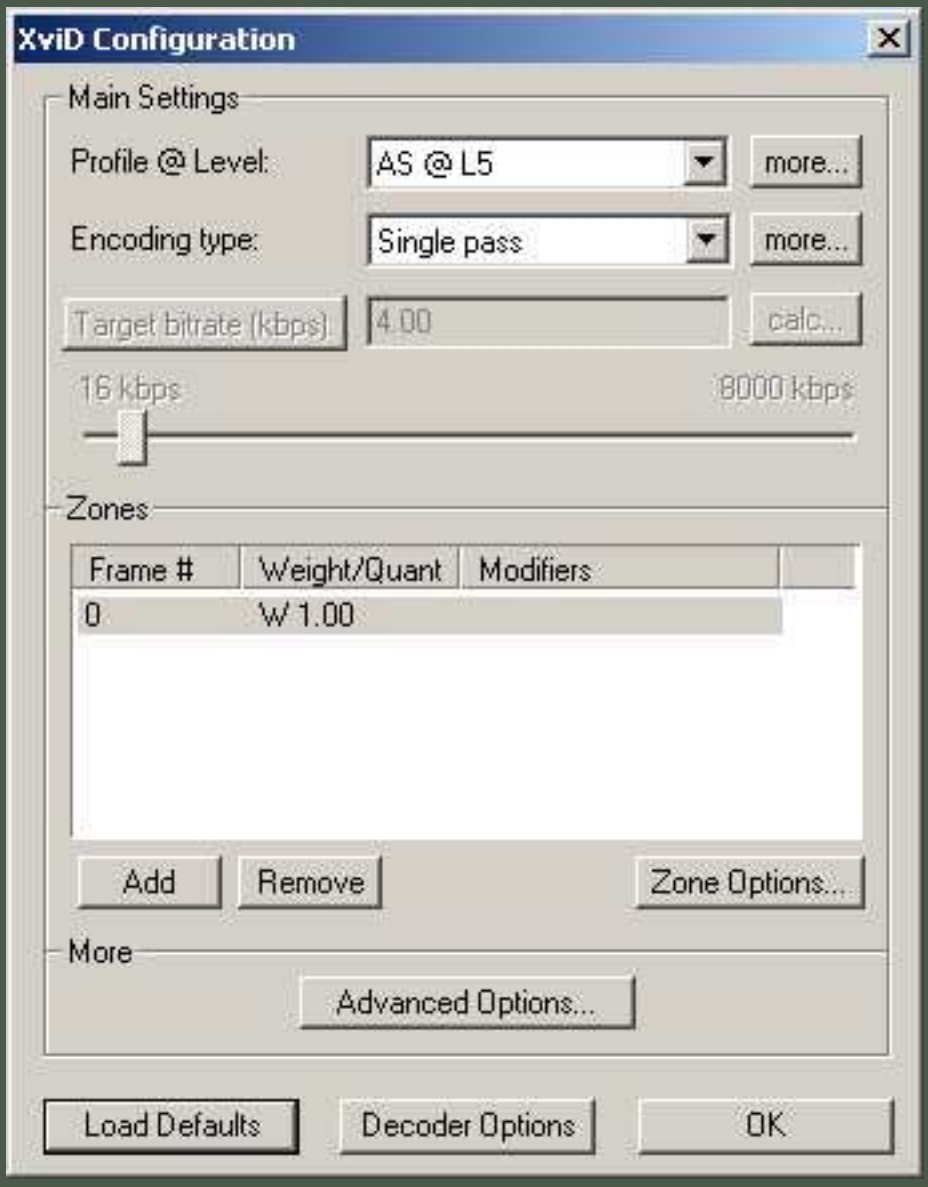
Part C: Options, Schmoptions....

Back to top

C1. What can you tell me about options in general?

When you open the XviD configuration tab in a program like virtualdub for the first time, you will see all kinds of buttons,sliders, subsections and subpanels featuring options, options and.... guess what? More options.
Basically, all the options are what makes XviD so much different from most other MPEG-4 codecs. They can make (or break) your encode and can also make a huge encoding speed difference. You can choose certain options that will make your result playable with codecs/decoders other than XviD (like for instance the DivX 5 or 3ivx codec or ffdshow) , or you can choose all those extra options that make XviD so unique and versatile. You can choose options that will allow for realtime encoding of a running TV- or Camera-capture, or you can choose options that will make the codec slow down to a crawl but will give really great results.

(The following description is for the Vfw(Video For Windows)-interface for the RC4-build. In other builds the interface could be different, but you will find many if not all options present in one way or another)



The Main configuration panel

The Main configuration panel has the subsections 'Main Settings', 'Zones' and 'More' (Note: 'More' is '...' in the betas). In 'Main Settings' You see 'Profile @ Level' , 'Encoding type', a button called 'Target Quantizer' and a slider ranging from 1 to 31 for target quantizer or 16 to 8000 kbps for target bitrate. Profile@level and the number of passes have to be set.
In the subsection 'Zones' you can assign different parts of the clip you want to encode unique zones, and set specific options for them with the 'Zone Options'-button. The 'More' subsection holds the 'Advanced options'-button which will pop up another configuration tab containing.... more options of course. :)

Beneath these subsections you find three buttons called 'Load Defaults', 'Decoder

Options' and 'OK'.
The first one will reset all options to their original default settings, the second button will allow you to tweak the decoder's post-processing settings. 'OK' is for when you're all done configuring and you want to switch back to the videotool (in this example virtualdubmod, but it doesn't have to).

C2. What are all these profiles about?

A *Profile* basically corresponds to certain MPEG-4 standards meant specifically for certain use-scenarios.
In each Profile you have different levels, which limit that particular profile to a specific bitrate-scenario. A bitrate scenario can set maximum limitations to bitrate, framerate, framesize and other settings. For example, Simple@Level 2 limits you to an upper bitrate of 128 kbps and a maximum framerate of 15p/s. Both Profiles and levels make hardware support easier because it defines certain maxima that have to be supported. Note: The DivX hardware profiles are roughly equivalent to this. In fact some betas of XviD had DivX-profile-compliant-profiles. ;)

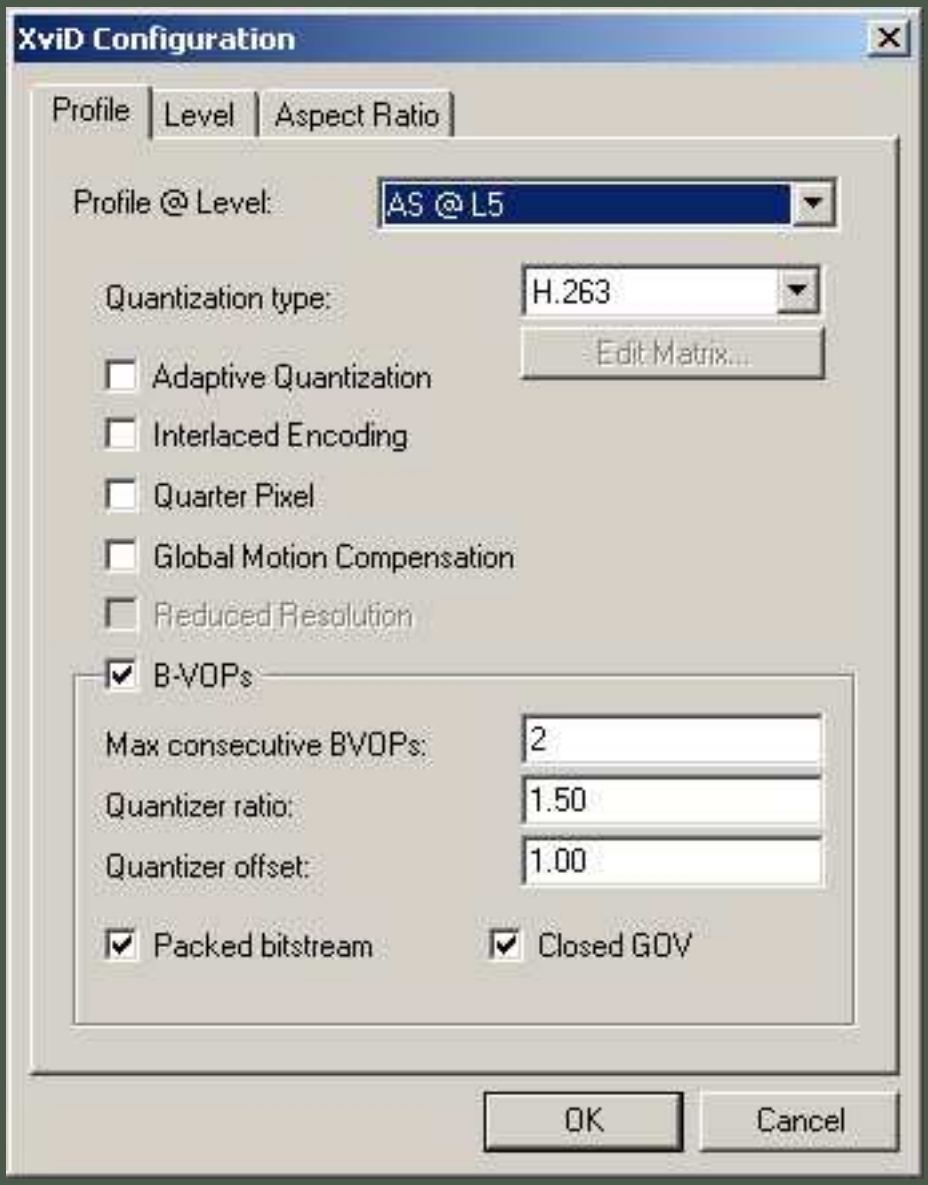
The 'Simple' profile is exactly what it says, 'simple'. It's fully equivalent and compliant to the MPEG-4 standard 'Simple' profile. It has many major limitations, not only on resolution and framerate but also on the kind of encoding options and optimizations you can use. It's primarily meant for low quality, low bitrate solutions (like mobile applications). Use it only if you have a specific need for it, like you need playback of your clips on a hardware device that only supports MPEG-4 Simple profile.

ARTS stands for 'Advanced RealTime Simple' and is basically the 'Simple' profile optimized for streaming applications. One extra option only allowed with ARTS is the 'Reduced Resolution' option, which is next to useless for most people anyway. It basically does nothing with the resolution but reduces detail to 1/4 of the orginal in certain cases. Use ARTS only if you have a very specific need for it.

AS stands for 'Advanced Simple' and is the most often used profile. It allows for nearly all available options to be used and allows for resolutions up to and including DVD-resolution of 720x576 at 30 fps, which is good enough for most people. It also allows for all sorts of 'advanced' options (not te be mistaken for the options in XviD's 'advanced options' tab, they are **not** the same). These include among others B-frames, Qpel, GMC, and also the use of other Quantization Matrices than H.263. Simple and ARTS only allow for H.263 to be used, whereas AS gives you the MPEG QM and MPEG-Custom, with which you can define your own Custom Matrix or load someone else's.

Finally you have the 'Unrestricted' profile that will give you unrestricted access to all options, resolutions and framerates. So if you want to experiment with a grayscale interlaced Cartoon with RRV at 2048x1024 @120 Fps, there's your chance. (Just don't expect it to work on anything else)

C3. Profile@Level "More..." panel - Profile tab



Here you can set options ranging from top to bottom:
-Profile@Level setting
-Quantization Type
-Adaptive Quantization
-Interlaced Encoding
-Quarter Pixel
-Global Motion Compensation
-Reduced Resolution
-B-VOP's
-B-VOP fine tune settings
-Packed Bitstream
-Closed GOV

C3a. Quantization Type: What is a Quantization Matrix?

To quote the explanation here:

The quantization matrix is the 8 by 8 matrix of step sizes (sometimes called quantums) - one element for each DCT coefficient. It is usually symmetric. Step sizes will be small in the upper left (low frequencies), and large in the upper right (high frequencies); a step size of 1 is the most precise. The quantizer divides the DCT coefficient by its corresponding quantum, then rounds to the nearest integer. Large quantums drive small
coefficients down to zero. The result:
many high frequency coefficients become zero, and therefore easier to code."

To put it short: It's the lossy filter by which you achieve higher compressibility by

losing detail. The amount of detail lost is determined by the values of each individual quantum.

More theory about the technical process behind it:
http://www.ece.purdue.edu/~ace/jpeg-tut/jpegtut1.html
http://www.mpeg.org/MPEG/MSSG/tm5/Ch7/Ch7.html
http://www.cmlab.csie.ntu.edu.tw/cml/dsp/training/coding/jpeg/jpeg/encoder.htm
http://www.acm.org/crossroads/xrds6-3/sahaimgcoding.html
Note: Most of these links are about Jpeg but the process is nearly the same for MPEG-4

C3b. Which quantization matrix should I use?

It is best to think of the MPEG quantizer as "sharpening" the image, and the H.263 quantizer as "softening" it. For high bitrates, you will find that MPEG quantization preserves more detail - for low bitrates, the smoothing of H.263 will give you less block noise.
You can also choose 'MPEG-Custom' which will allow you to choose other Quantization types of your own choice. When you do, the 'edit matrix...' option will become active which will allow you to modify, load or save a custom Quantization Matrix. Custom matrices are considered 'for advanced users only' so as a newbie it is best to stick with the two default choices, which are pretty good anyway.

C3c. What is Adaptive Quantization/Lumi masking?

Adaptive quantization, called 'Lumi masking' in older builds, is a first 'psychovisual' innovation in XviD; it is supposed to make use of the fact that the human eye tends to notice encoding errors less if they happen in very dark or very bright parts of the picture. XviD is capable of using different quantizers for each macroblock. Lumi-masking compresses very dark or bright areas stronger than medium ones. So it will use less bits on some frames in the second pass than in the first pass. The saved bits are of course spent again and that way we gain a bit of quality in the medium-brightness part of the picture. As it is experimental, you may sometimes notice more blocks than when it is disabled. Note that Lumi masking is broken in Koepi's 26062003-1 build, and that it is a bit buggy in many older builds. AQ works in the newer beta and RC builds but is still not very extensively tested.

C3d. Interlaced Encoding

This allows you to encode the entire clip in interlaced mode. Note however that your source has to be genuinely interlaced for this to give the proper results. Most interlaced video is *not really interlaced* but mutilated progressive content. Especially most interlaced DVD's and VHS video's are (sometimes poorly) interlaced versions of (progressive) film.
Typical examples of true interlaced content are live TV-shows and Camcorder footage.

C3e. How does Q-Pel work and when should I use it?

Q-pel (or Qpel) is the short name for **Q**uarter **P**ix**el** motion search precision and this option activates the use of quarter pixel precision.

Motion search tries to capture all the motion between one frame and the next, so that the macroblocks (MB's) can get the right motion vectors assigned to them. If the motion is properly captured then there will be no need for extra alterations to the MB other than a motion vector, sparing quite some bits. The more precisely the motion is captured, the less bits have to be assigned to the content of the MB's, and the more MB's can just consist of a motion vector.
So, *theoretically*, a more precise motion capture would save in altered texture information, thereby saving bits, and increase precision of the overall compression, thereby increasing quality. (We will soon see why this is just *theoretically*)

Normally XviD uses half-pixel motion search precision. This means that it can 'see' movement in a sub-pixel precision; if a MB moves from a width,height-position of 200,300 to 201, 300 in the next two frames, it can detect that movement correctly and can give the MB a motion vector that says "move me half a pixel to the right this frame please" in those next two frames. Motion will be captured correctly and no texture bits get altered.
Now with Qpel you can capture motion that is only a quarter of a pixel per frame, effectively **doubling precision**.
Example:
A MB that moves (fluently) from position 200,300 to 201,300 in the next **four** frames only moves one quarter of a pixel per frame. With normal half-pixel precision this motion would appear 'jumpy' and the codec might have to compensate for this *by altering the texture bits of the MB*. This of course takes space, and the MB would no longer consist of only a motion vector; it would have to be assigned additional bits for the altered texture information, thereby decreasing compressibility.
With Qpel that motion still gets captured correctly and not extra texture bits would have to be assigned, reducing the number of bits used for that frame.
Easy eh? But wait, there's a catch....

So, what's the catch?

The catch is, that just using the Qpel precision alone already uses additional bits, **whether it helps saving bits or not**.
This is caused by the additional precision that requires more bits to be allocated to the motion vectors. Instead that a motion vector could just be something like 0.5,0 (half a pixel width movement, no height movement) it would no have to be 0.25,0 (quarter of a pixel width movement, no height movement). So instead of one decimal after the point it now requires two decimals behind the point, requiring the codec to throw more bits at it.
(Please note that this is an oversimplification of the actual process, but it is accurate

enough to get the point)
Instead of another decimal Qpel actually uses another extra bit (set either to 0 or 1) for every axis, which is enough to achieve double precision. There are two axes, one for width and one for height, so each motion vector requires two extra bits for Qpel.
If we assume that there is one vector for every macroblock (there might be 4 or 0), at the resolution of 640x272 and 24fps and P-frames only, two bits for every macroblock take 40 x 17 x 2 x 24 = 32640 bits or **32.5 kbps**.
So, basically, no matter the outcome, Qpel always takes a sizeable chunk of the bitrate just for itself *even if it doesn't help compression one damn bit*.
Now usually it does help, but the texture bits saved by the better precision **have to outnumber** the added bits to the motion vectors before Qpel **increases compressibility at the same size**. If the saved texture bits outnumber the extra motion vector bits then you will have increased compressibility (and quality) at the same size. If the saved texture bits **do not** outnumber the extra motion vector bits you will have wasted space and the end result might look worse.

So how can I tell if Qpel will increase or decrease compressibility?

That's the other catch: **You can't**. Not in advance. There's no way to tell from looking at the source if Qpel will help or not. It doesn't matter if it's a fast motion scene or a low motion scene, a panning scene or a zooming scene...there's just no way to tell beforehand. A fast motion scene could be 90% Qpel movement or 90% Half-pixel movement, or any percentage in between...making any prior assumptions about the benefits of Qpel ridiculous.

The only real way to find out is to try encoding both with and without Qpel and see which result looks better.
(Now you can see why there is a difference between theory and practice...)

If you like more information read the discussion in this Doom9 forum thread
Some additional Notes:
-Because of the increased precision, Qpel significantly increases encoding time, and requires more processing power to decode. Encoding time can be almost doubled and decoding can require as much as 30-60% more processing power.
-Current 3ivX implementation normally uses half-pixel precision. As an 'advanced' option, you can tell it to use *full-pixel* resolution.
-In some older (alpha) builds Qpel could introduce artifacts, but the current implementation has no known bugs. It's safe to use.

C3f. What's GMC and what is it good for?

GMC stands for **G**lobal **M**otion **C**ompensation and that pretty much tells the story of what it does. If used, it will look at the whole frame and see if there is an amount of motion that all the parts of the frame have in common. It will then take this amount of motion and put it in a single value. The parts of the frame are the macroblocks, and the amount of motion is called a 'motion vector' which has both a direction and a value (as

a sort of two-dimensional X,Y value) .

All the macroblocks normally have their own motion vectors, but with GMC the one motion vector that they all have in common (that's why it's called 'Global') will be compensated and put into a single motion vector. Some macroblocks' movement will be completely compensated for by the GMC vector, getting completely nullified by the compensation process. These macroblocks' motion vector will then be removed, as it is the same and is only extra information. The possible benefit is that you can remove many or all the motion vectors of the macroblocks (or even the blocks themselves if there is no altered texture information) in a frame by a single value, thereby making it much smaller.
Note however that this is for one-warppoint GMC. With Multiple warppoints the proces is much more complex, but the principle is the same.

C3g. Warppoints, hmm...what's a warppoint?

A warppoint is a motion vector that defines a displacement of one *edge* of the video. Take a piece of paper and move it by its edges and you'll see what I mean.
-The first warppoint defines displacement of top-left edge. If it's the only warppoint, the rest of the picture just has the same vector and the whole picture moves. Think panning.
-A second warppoint defines displacement of top-right edge (not *precisely* true but close enough without getting too technical). Together with the first warppoint, this is enough to define panning *and* zoom. Note that it could be used to define panning and rotation instead, but isn't.*
-A third warppoint means displacement of down-left edge and three warpoints are enough to define panning, zoom and rotation.
-A fourth warppoint would create perspective-like movement.
Note that XviD's GMC uses 3 warppoints, whereas DivX's GMC uses only one. Warppoints are stored in the frame's header, and only if they are used.

C3h. What is this Reduced Resolution ?

Reduced Resolution (also called RRV for **R**educed **R**esolution **V**OP) is in fact reduced resolution. Macroblocks become 32x32 pixels big, and all DCT data - once decoded - is scaled up to be 16x16 (DCT itself is still 8x8). Motion is also severly restricted, with vector components restricted to either zero or odd values (odd values point to halfpixel positions). Also, an in-loop deblocker kicks in. Everything described in the standard.

RRV is part of the realtime profile and only works with the ARTS and unrestricted profiles in XviD. It was designed to keep the picture "visible" if bandwidth can suddenly become very limited. Reduced resolution VOPs are dynamic and can be turned on and off at any time, but XviD does not offer this - at least not by VfW.

I don't see any good reason to use RRV, because if the decision is not dynamic (like now in XviD), you'd be better off if you just encoded with halved resolution. Also,

XviD is the only decoder that can play such files (90% of MPEG-4 decoders can't). The only realistic use one could possibly have for this feature is for encoding for low-bandwidth mobile devices using GPRS/UMTS.
It's still not fully mature and it is to be used only by adventurous people.

C3i. BVOPs: B-frames

B-frames (or BVOPs in techie talk) are so-called bi-directionally encoded frames and are part of the Advanced Simple Profile definition. Without B-frames you just have the Keyframe making a clear definition of the content of the frame every XXX frames, and all other frames are P-frames referring to the previous frame for description. A B-frame also takes the next frame into account, so it refers to other frames in two directions (ergo the bi-part). The advantage of B-frames is that they are usually encoded with a higher quantizer and take less space per frame, while the loss in quality is less than equal to the loss in used bits. Basically you use the inherently smaller and lower quality b-frames to save space that will be used for improving quality all over the clip. The netto effect is usually a quality gain, depending on the b-frame settings and the type of source.

C3j. B-frame settings

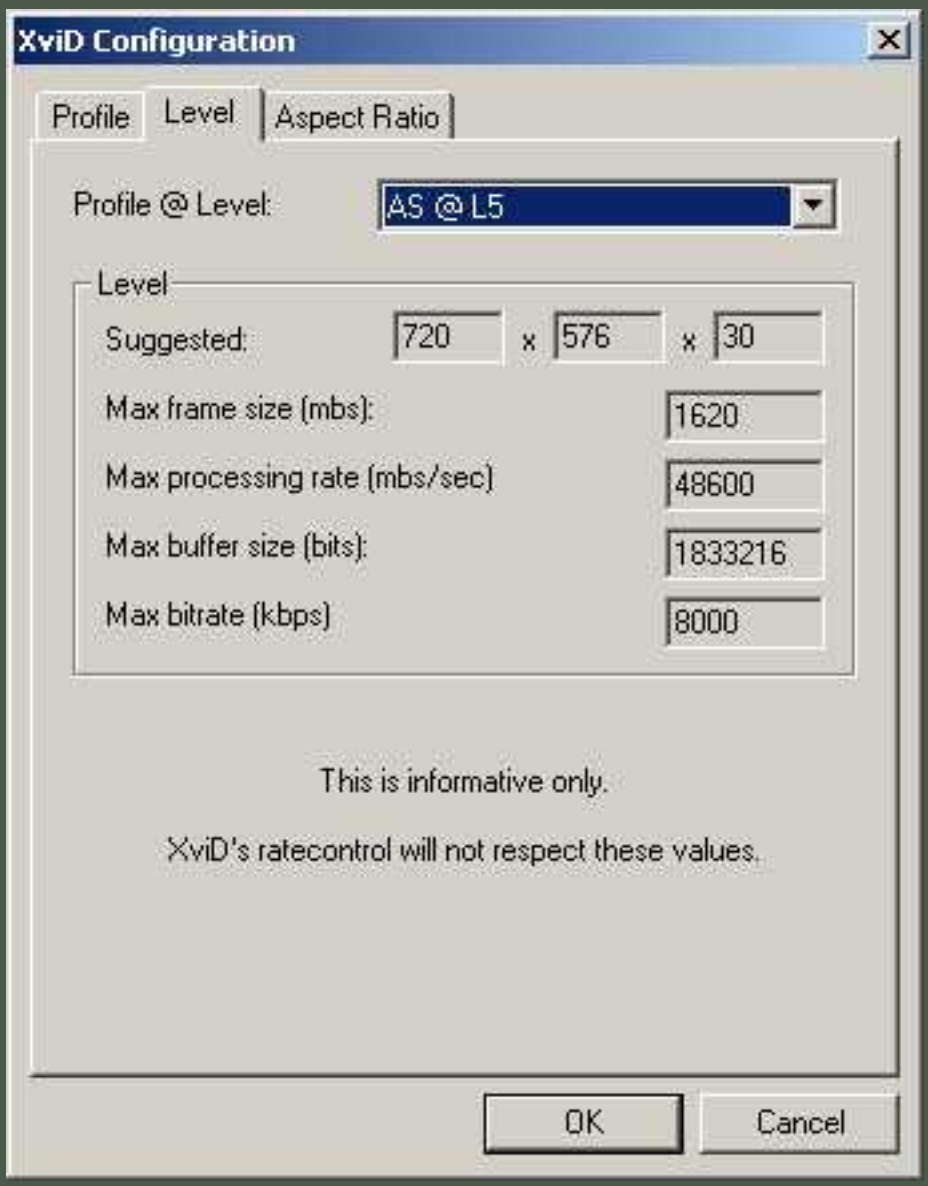
-Max Consecutive BVOPs: Here you can limit the number of B-frames in a row. Recommended settings are 0 for off, 1 for DivX 5 compatibility, 2 for best effect and 3 for intensive use.
-Quantizer ratio: Multiplying the (average) quantizer of the surrounding non-B-frames with this value will give you the Quantizer of the B-frame. So if the two adjacent frames have quantizers of 2 and 4, the average quantizer will be 3. Multipying this with a quantizer ratio of 1.50 will give you a B-frame with a quantizer of 4.5.
-Quantizer offset: Take the result of the calculation above and then add this value. With a quantizer offset of 2.00 you will end up with a quantizer of 6.5.
As a rule of thumb, upping the latter two values will give you lower quality B-frames.

C3k. Packed Bitstream and Closed GOV

-'Packed Bitstream' is an option that can deliver mixed results during playback, depending on what you use for playback. It's meant to solve frame-order issues when encoding to container formats like avi that can't cope with out-of-order frames. And while it's meant to solve playback issues that occur without it, lots of people have reported playback issues **with** it. That goes for playing back with ffdshow, DivX 5 decoder, and several standalone (hardware) players.
Unless you know *precisely* what you're doing, it's best to keep it turned off until further notice. If you encounter problems with choppy playback, try turning this feature on and see if it helps.
Note: If you only play your files with the XviD codec, you **never ever** have to use packed bitstream.

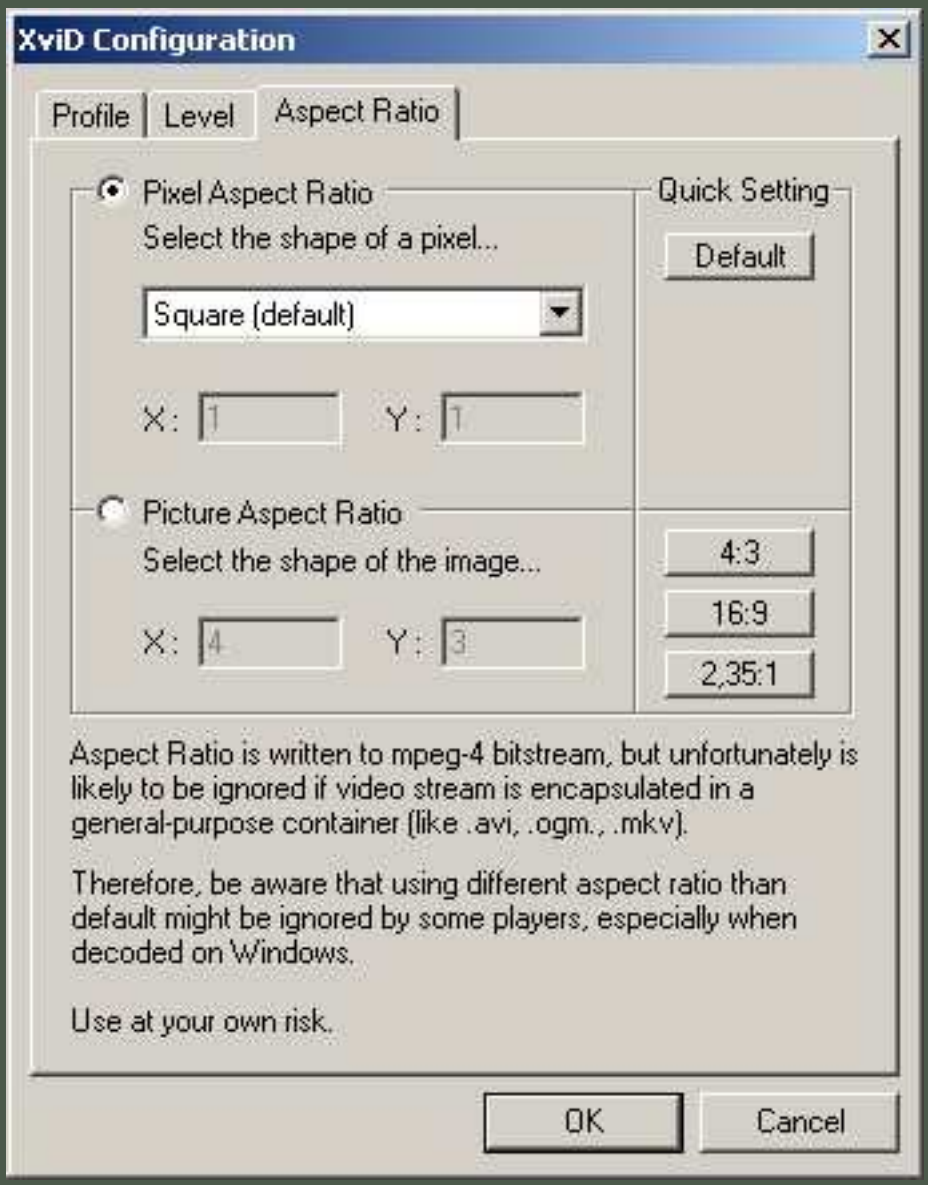
-'Closed GOV' (Quote from Doom9):
Closed GOV ensures that a p-frame is used before every new I-frame. This option should always be checked (otherwise you might end up with a frame sequence like PBIP where the B frame has a forward reference to an I-frame which makes no sense).

C4. 'More' Profile options - Level tab



This tab is informative only and shows you the different bitrate limitations of each profile@level setting. You can clearly see that maximum rates go up with each higher level. It should be no surprise then that AS@Level 5 is currently the recommended setting. 'Unlimited' is exactly what it says, and is there for more adventurous people who want to experiment with all the available features. Please also realize that right now the actual limiter in the codec itself is **not implemented yet**, as it's hard to do properly. You have to check for yourself if your file adheres to the limits defined by the profile you chose.

C5. 'More' Profile options - Aspect Ratio tab



-'Aspect Ratio' **(AR)** (sometime called **DAR**, for **D**isplay **A**spect **R**atio) is the ratio between height and width of the picture. In general you can use any AR you'd like, but for most encodes that have something to do with encoding from a standard format (like TV, DVD or VCD) or decoding to one, you can use this tab to select an AR for the entire encode or an AR for the pixels (PAR).
Why you would want to do this? To properly capture the original resolution of the video you want to encode. Consider a 720x576 DVD. With a square pixel that DVD would have quite a different look when played than if the pixel was of a standing or lying oblong size.
And yet that is what happens when you convert from NTSC to PAL, from DVD to VCD or from TV to computer. NTSC, PAL, VCD and computer monitors all have different pixel sizes, and some formats that where created before the digital age (like all TV standards) have different ways in which they treat pixels.

In reality however, the difference between them generally is small and few people have noticed the difference. Also, support for proper handling of the AR information in clips is still very rare. So while this option is here to enable the purists among us to completely recreate the original AR, you don't have to use it if you don't want to. In fact, it's full and correct implementation requires a rather thorough understanding of the underlying problems of different Video AR standards, and a discussion of these problems is **far beyond the scope of this faq and you should consider the AR tab for advanced users only.**

If you want to know more about proper AR implementation, follow these links:
Doom9 forum discussion about this feature: New Feature: Display Aspect Ratio
A document outlaying the differences between pixel sizes: Square and Non-Square Pixels

A document outlaying the differences between video AR standards: A Quick Guide to Digital Video Resolution and Aspect Ratio Conversions

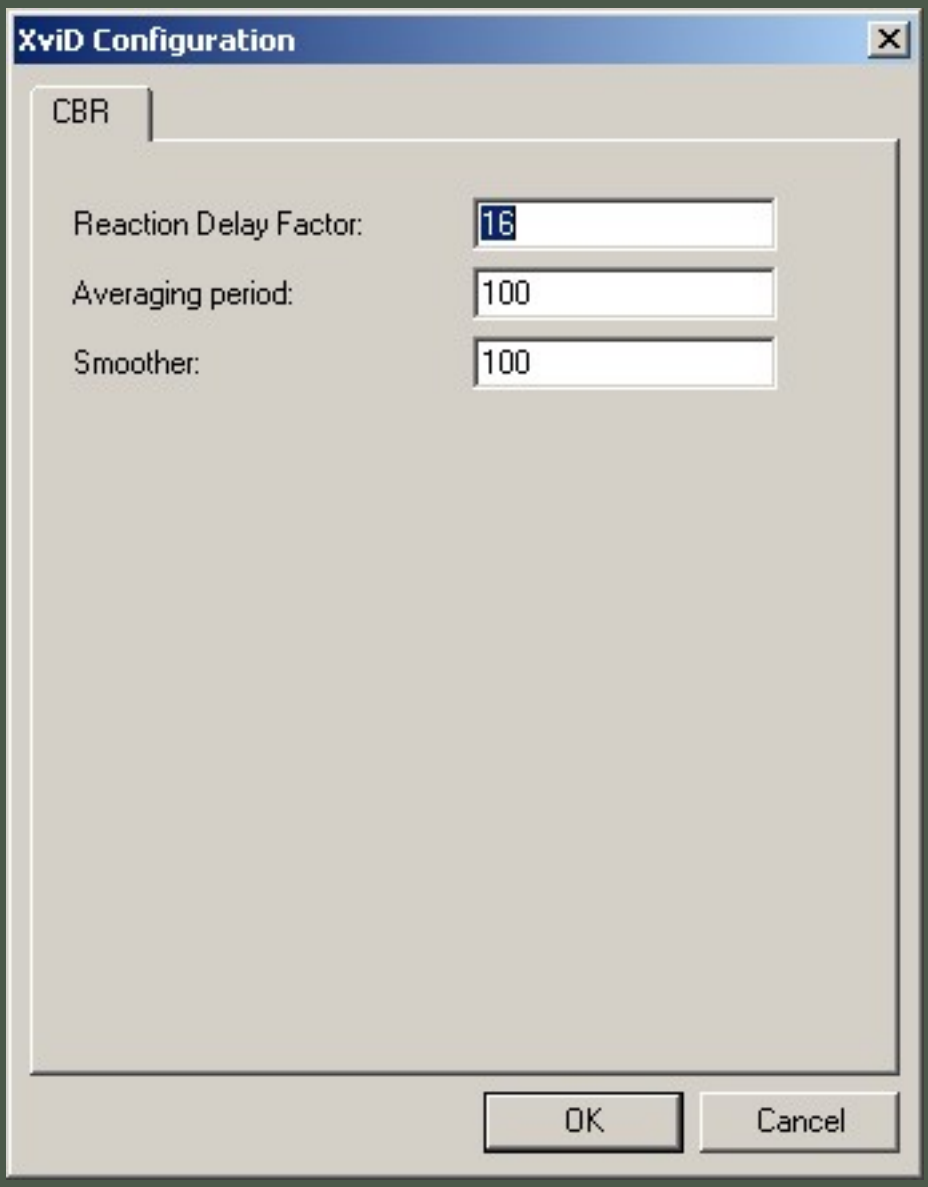
C6. Encoding type: Single pass or two pass, what should I do?

-Single pass will take your clip and encode it at once. It takes each frame of the clip, checks that frame's compressibility, and then encodes it.
-Two-pass uses the first pass to make an estimation of how well your clip compresses and then uses the compressibility data gathered during the first pass to really encode the clip during the second pass.

Which one to choose depends on what you desire from the result. Two-pass does a much better job at evenly distributing bits where they are needed and therefore gives you a much better looking end result. Single pass is really for those type of uses that can only be done with single-pass, like for instance real-time encoding a live feed, like a TV-capture or a security camera. Unless you absolutely have to go for single pass for a specific reason there really is no other way but two-pass.

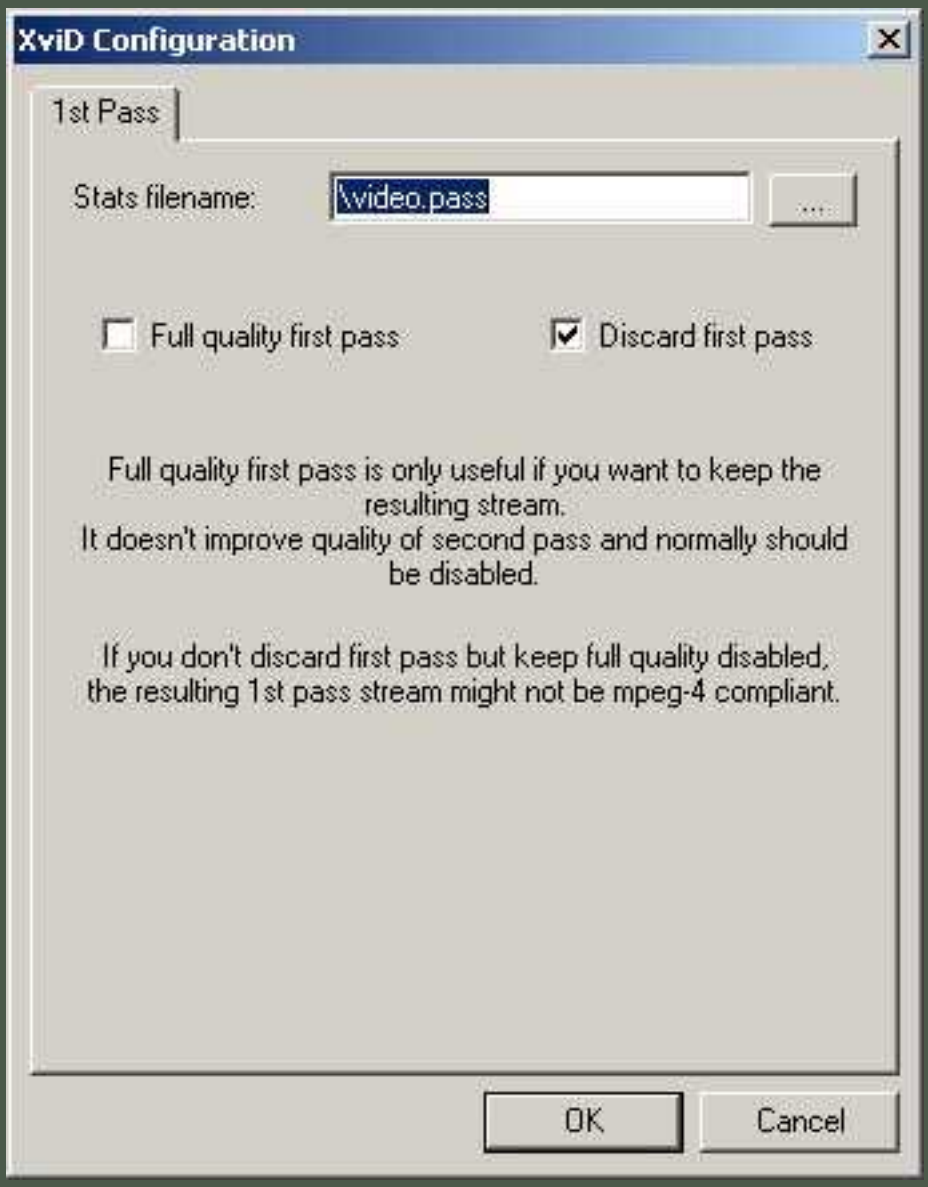
Note that DivX 5 nowadays has a 'multi-pass' option, allowing for more than two-passes. This is meant to tweak the bit-distribution even more (sort of by averaging between a large number of passes), but many users report nearly zero benefit after the third pass. XviD really doesn't need a technique like this because the bit-distribution-decision it makes is more intelligent and produces better results.

C7. Single pass 'more' settings explained



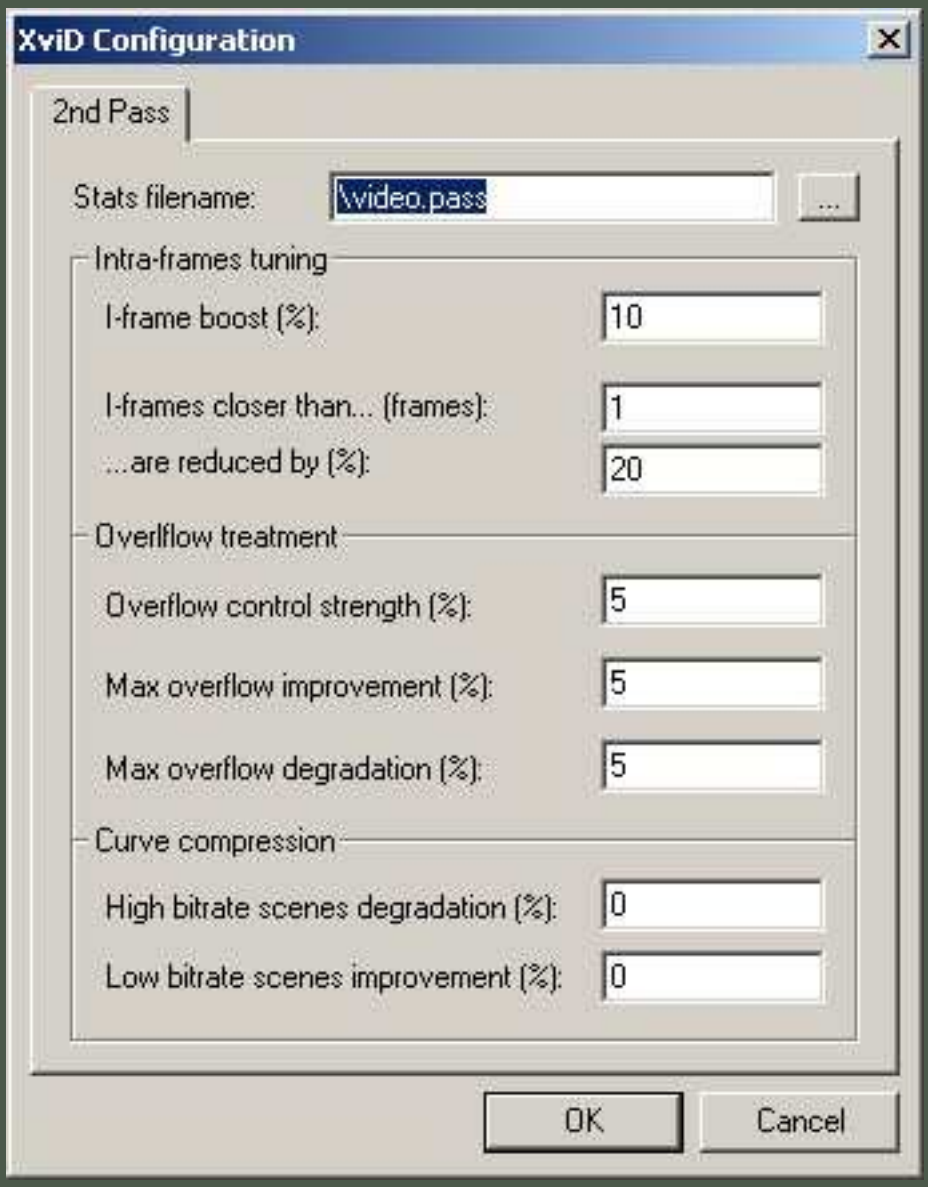
Work in progresssssss....

C8. Two-pass first pass 'more' settings explained



Here you can alter three settings: One is the name and path of the stats file (this is the data file with the compression data gathered by the first pass).
-'Full quality first pass':By default XviD switches certain options off in the first pass to speed it up a bit. The options that are switched off are not really necessary for a 'normal' first pass (normal as in: you don't keep it afterwards) and turning them off can increase encoding speed considerably. However this is a relatively new option and some people are concerned that the second pass *might* have a little bit less quality because of the options turned off in the first pass. Since the jury is still out on this one you can set this option to make the codec use all the options for both passes, if it makes you feel comfortable. Recommended setting is **off** so the codec can speed things up a bit.
-'Discard first pass' determines whether or not the first pass should be kept or not. It is a proper video file, and you can play it but it is not recommended that you keep it. You should think of it as no more than a rough approximation of the end result and even though you can play it, it might not be MPEG-4 compliant. Always discard it unless you turned on 'Full quality first pass'. (Then you may choose to keep it, as the file created by the first pass will then be completely normal)

C9. Two-pass second pass 'more' settings explained



Most of these settings are meant for tweaking of the allocation of bits during the 'real' encoding pass. They are for advanced users only and are best left alone.

-The first setting here is the name and path of the stats file to be used. Normally you should keep this the same between first and second pass. of course you can experiment with this by using the stats file from a pass with other options, but for normal encoding it's best left alone.

Intra-frames Tuning

'I-frame boost(%)' can be used to give some extra bits to Keyframes. The amount is an extra percentage, so a value of 10 will give your keyframes 10 % more bits than normal. (Note: I know of *no circumstance* that would justify setting this to any other value than default)

-'I-frame closer than ... frames' and '... are reduced by %' can be used to adjust the size of keyframes that you consider to close to the first (in a row). The first setting sets the range in which Keyframes are reduced, and the second settings determines the bitrate reduction they get.

The last i-frame will get treated normally.

Overflow treatment

'Overflow treatment' is the technique used to obtain **a properly sized end result**. Usually you specify a target filesize and the codec can either *overshoot* it's target, creating a file too big, or it can *undershoot* it's target, creating a file too small. Too counter this, overflow treatment can either allocate more bits than abolutely necessary, increasing filesize, or allocating less bits than really necessary, decreasing filesize. Obviously, the second process involves compromising quality.

-'Overflow control strength %'

This is best described as the *'aggresiveness'* of the overflow control.

Mental picture: Imagine the 'Overflow control mechanism' as a little guy in your computer running around with **glue** and a **chainsaw** redistributing bits all over your

clip. Now imagine him getting more agitated. :^)
Higher settings will make changes in bit redistribution more abrupt, possibly too abrupt if you set it too high, creating artifacts.
0=Default from core (let XviD decide). I know of **no obvious reason** to experiment with this setting.
-'max overflow improvement %' sets in % how much each frame *may* **grow** if the file would become **undersized**. Think of it as **'adding fat to your clip'** if it's not as fat as you asked it to be.
-'max overflow degradation %' sets in % how much each frame *may* **shrink** if the file would become **oversized**. Think of it as **'trimming fat off your clip'** if it's a bit too far on the porky side. :^)
Obviously, experimenting with these settings **may break filesize prediction completely** as you're altering the basic settings of the underlying process.
(Personally, I couldn't care less about the first setting because I consider undersized files to be a **good** thing)

Curve compression:
Normally the internal curve adjustment values (determined by the XviD developers after much feedback from users) are capable of delivering very nice results (I should say 'excellent' really), but if for one reason or another you want to, you can use these values to adjust the lows and highs of the bit allocation.
If you make a mental image of the curve allocation, you see a graph with 'highs' and 'lows', sorta like hilltops and valleys. The hilltops are scenes with high bitrates and the valleys are scenes with low bitrates.
-The 'High bitrate scenes %' setting will **take bits away from high bitrate scenes** and give them back to the bit reservoir. (Think of a *bit reservoir* as a bucket full of bits wherefrom the codec hands out to each frame) Ergo, it will lower the hilltops, and the bits gained by this will be divided equally across the entire landscape. This is useful if you really need to keep your encode within certain maximum parameters, like the maxima for a specific profile@level setting.
-The 'Low bitrate scenes %' setting will **give extra bits to the low bitrate scenes**, sort-of-like filling the valleys with sediment. The bits have to come from somewhere, so the codec takes all the frames in the entire encode and scrapes a few bits of them all. This might come in handy if you have a few low-bitrate scenes that are still blocky.

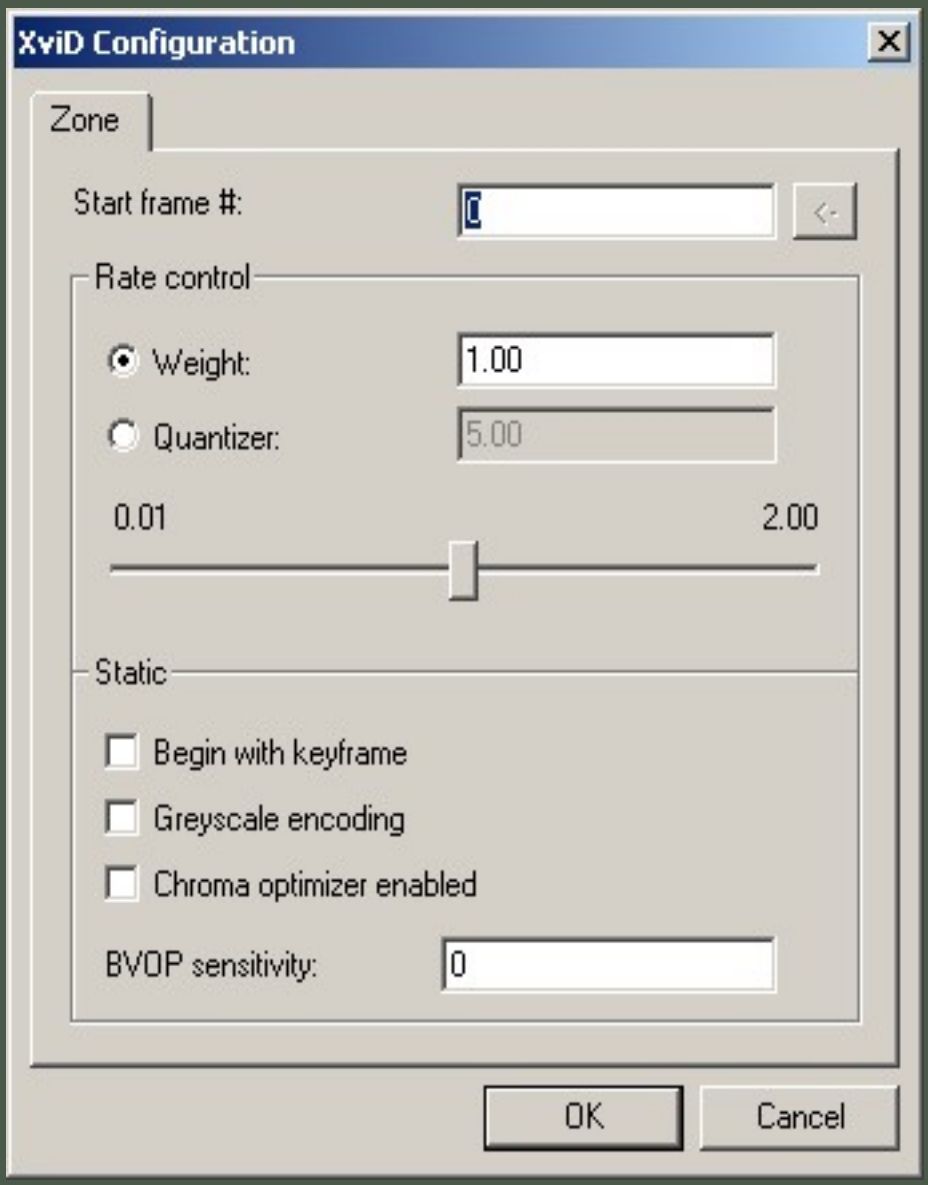
So, basically, *each setting favors compression of one of two possible extremes towards the average of the entire encode*. The first takes down the hilltops, the second fills the valleys. The bits lost or gained are allocated accordingly to **average out** the entire clip.

C10. What's this 'target quantizer' and what I should I do with it?

The 'Target Quantizer' button allows for different things depending on the number of passes you selected. In 'Single pass' it allows you to switch between 'Target Quantizer' and 'Target Bitrate'. The input field next to it and the slider below alter accordingly showing the possible settings.

In 'Twopass - 1st pass' mode the button, input field and slider are inactive. This is logical since the first pass is basically an expedition to gather data about your clip, without making any real decisions on the resulting file. In 'Twopass - 2nd pass' mode you can choose between 'Target Bitrate' and 'Target Filesize' instead of 'Target Quantizer'. This reflects the fact that most twopass encoding scenarios are aiming for a specific filesize to be reached.
Please note that filesize is in Kilobytes not -bits or Megabytes. So if you're aiming for a file of 35 MB you have to enter 35840 (35x1024) instead of 35.000.
Also, when you set the button to 'Target Bitrate' you also have to keep in mind that the maximum bitrate is to be limited by the Profile and level settings. Currently XviD does not do that automatically (mainly because it is difficult to implement) but you should be aware of the limitations of the profile and level you use.

C11. Zones?



Zones allow you to assign different settings to different parts of your clip. You can add as many zones as you want, and each zone can have it's own unique settings.

It's most useful function is to allow for certain parts of your clip to be considered 'less important' than the rest by defining zones and how 'important' they are to you. You can define this importance in the tab accessible through the 'Zone Options' button.

To assign specific settings to a zone, select it in the 'main' panel and then click on 'Zone Options...'.

There are two different ways to define the importance of a zone:
-Weight defines your 'importance' as a value relative to the quality of the whole clip. If you define a zone with a weight of 0.30 then that zone's bitrate will use 30% of the quality of the whole clip.

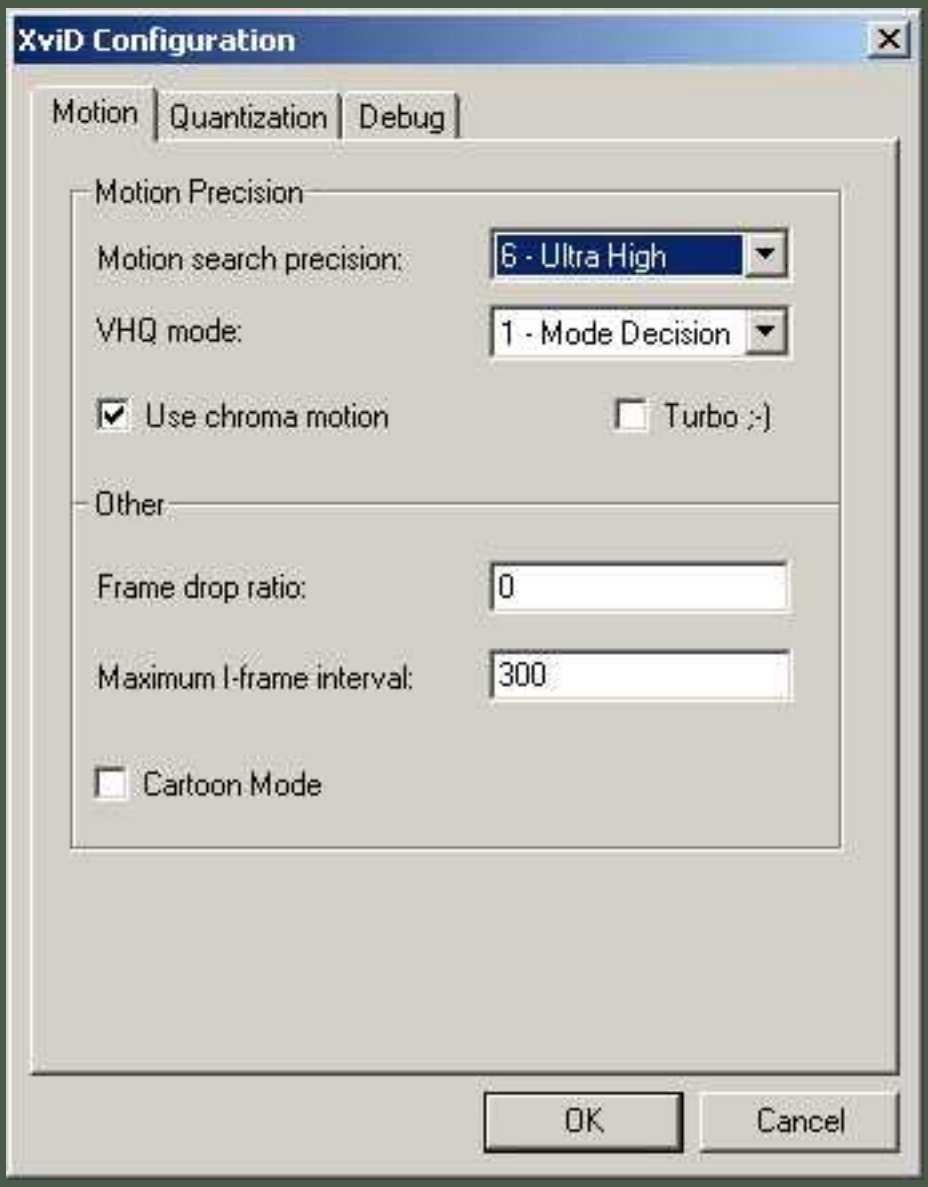
-Quantizer defines the 'importance' as an absolute average. What that means is that a Zone with a Quantizer of 5 will absolutely have an average quantizer of 5. How this saves bits depends of course on what settings the other zones have.

Other zone options include:
-'Begin with Keyframe' will make each zone start with a Keyframe. Recommended, especially if you plan on using chapters.
-'Greyscale Encoding' tells the codec to encode without color information. Great for saving bits on boring end credits.
-'Chroma Optimizer enabled' will do some extra magic on color information to minimize the stepped-stairs effect on edges. It will improve quality at the cost of encoding speed. It reduces PSNR by nature. The *mathematical* deviation to the original picture will get bigger - but the *subjective* image quality will raise (as mentioned, the "stair step artifacts" get less). Since it works with color information, you might want to turn it off when encoding in greyscale.
-BVOP Sensitivity will allow you to tweak the amount of B-frames in each zone. A positive value increases the amount of B-frames; a negative value decreases it. Currently the minimum is -35, anything below will not make any difference. There is no maximum, but value is pretty sensitive; 5 will give you a reasonable boost. I'd say that practical maximum is about 25.

It's advised to be conservative with zones, as it's still a relative new development, and pushing it's limits *might* give unexpected results.
Current bugs (Xvid-RC2): Weight zones with weight lower than 0.2 break filesize prediction.

The 'Advanced options' window

C12. 'Motion' tab



A) - Motion Precision

Here you will find the options defining the motion search precision. Motion search is the process in which the codec is trying to figure out how every part of the original clip was moving. The more it searches, the more precise it's estimation of the original motion will be, and the better the resulting clip will capture the original motion.

Why capture motion you might ask?

Let's take a look at a simple example, a clip of a white block moving to the right. Each frame a part of the picture where the block no longer is becomes the background color while another part has to become the block color. The change that would have to be encoded each frame would constitute a significant amount of bits.

Instead the codec just takes the block and checks to see if it is moving. If it moves, the codec captures that movement with it's motion search, and then uses the found value as a motion vector for that specific block. In reality the process is more complex but the basic idea is that most altering of color and texture information is caused by motion and therefore a great amount of color and texture bits can be saved if that motion is captured by other means.

-Motion search precision: This is the most basic search precision and it hardly takes any processing time, so it's recommended to leave it at 6. Only go as low as 5, and only if you're in a hurry. BTW: It only works in the luminance plane, i.e. it only looks at changes in brightness values, not color.

-VHQ mode: VHQ is more intensive search and takes a wider approach. Higher settings will slow down encoding significantly. Setting 1 has a relatively small impact and it is recommended for all encodes. Using higher values will give you better quality at the cost of encoding speed.

-Use Chroma motion: Chroma motion is basically a sort of 'Motion search precision setting 7' whereby it also takes color information into account (from the Chroma plane,

ergo the name). Recommended.
-Turbo ;-): This setting skips some search techniques when using Qpel or B-frames to speed it up a bit. Without those options on it has no effect at all. The impact on quality is negligible.

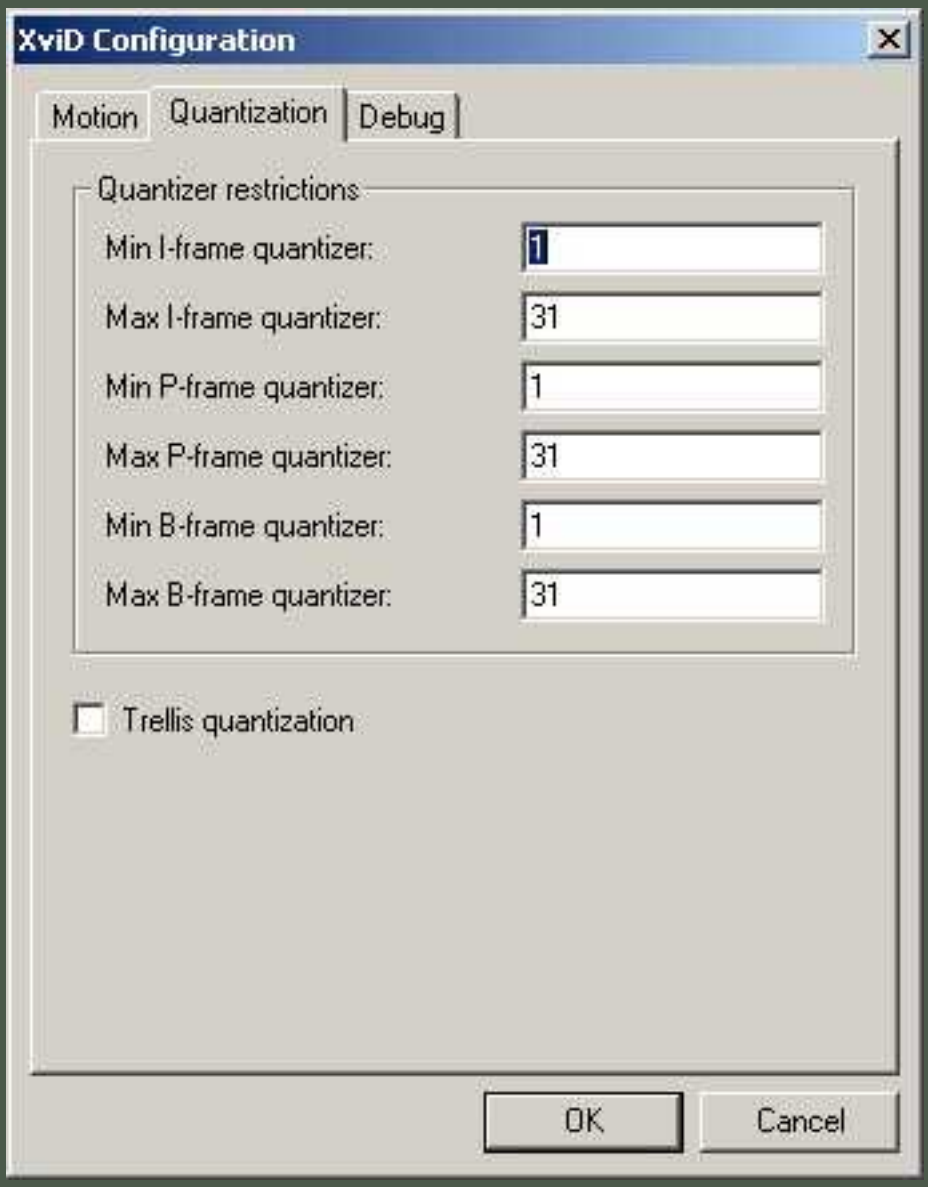
B) - Other
-Frame drop ratio: Experimental/For experts only. Don't set it to anything other than 0 unless you **really really** know what you're doing.

-Maximum I-frame interval: This setting tells the codec to insert a Keyframe (I-frame) every {value} frames. If a Keyframe is needed before that number is reached, the codec starts counting again. So while you can have Keyframes with lower intervals than the number you defined, you can't have higher intervals.
Standard recommended settings are 10x the framerate, i.e. 250 for 25 fps PAL clips, 300 for 29.979 NTSC clips etc. However, there is a visible effect called Keyframe-pumping. This resembles a slow degradation in quality in consecutive P- and B-frames with a sudden 'jump' in quality when a new Keyframe is inserted. Lowering the maximum I-frame interval in these cases can help. Setting it too high can cause bad seeking in files, because the seeking process uses only Keyframes, and less Keyframes=less accurate seeking.

-'Cartoon mode' activates two different techniques, both designed to help with cartoons:
- detect_static_motion is a motion estimation flag and it works as a limit. If the motion found by the motion search process is beneath this limit the macroblock is considered static, and **no motion** information is encoded. When Cartoon mode is enabled this limit below which a macroblock is concidered static is increased so that (more) small movements are lost. Since A LOT of these 'small movements' is actually noise (especially with cartoons) it really helps saving many bits which would otherwise be used to code noise on a static picture.
- vop_cartoon is about quantization - when a block is motion-compensated well enough (with total error below the limit) it's just not coded at all. XviD doesn't drop any data in normal mode (limit = 1), but drops quite a lot in cartoon mode. Again, this usually means that noise is ignored. It might also remove some small details, but small details shouldn't really happen in "proper" cartoons.
So, while the first technique helps with removing movements that are so tiny that they can be considered not-to-be-part-of-the-source, the second helps compressibility of the cartoon by removing texture detail that's considered too-small-to-be-part-of-the-source.

This is exactly what you need for cartoons like Futurama or Simpsons, but it *might* not be optimal though for high quality animes with lots of fine detail; try and see if it helps.

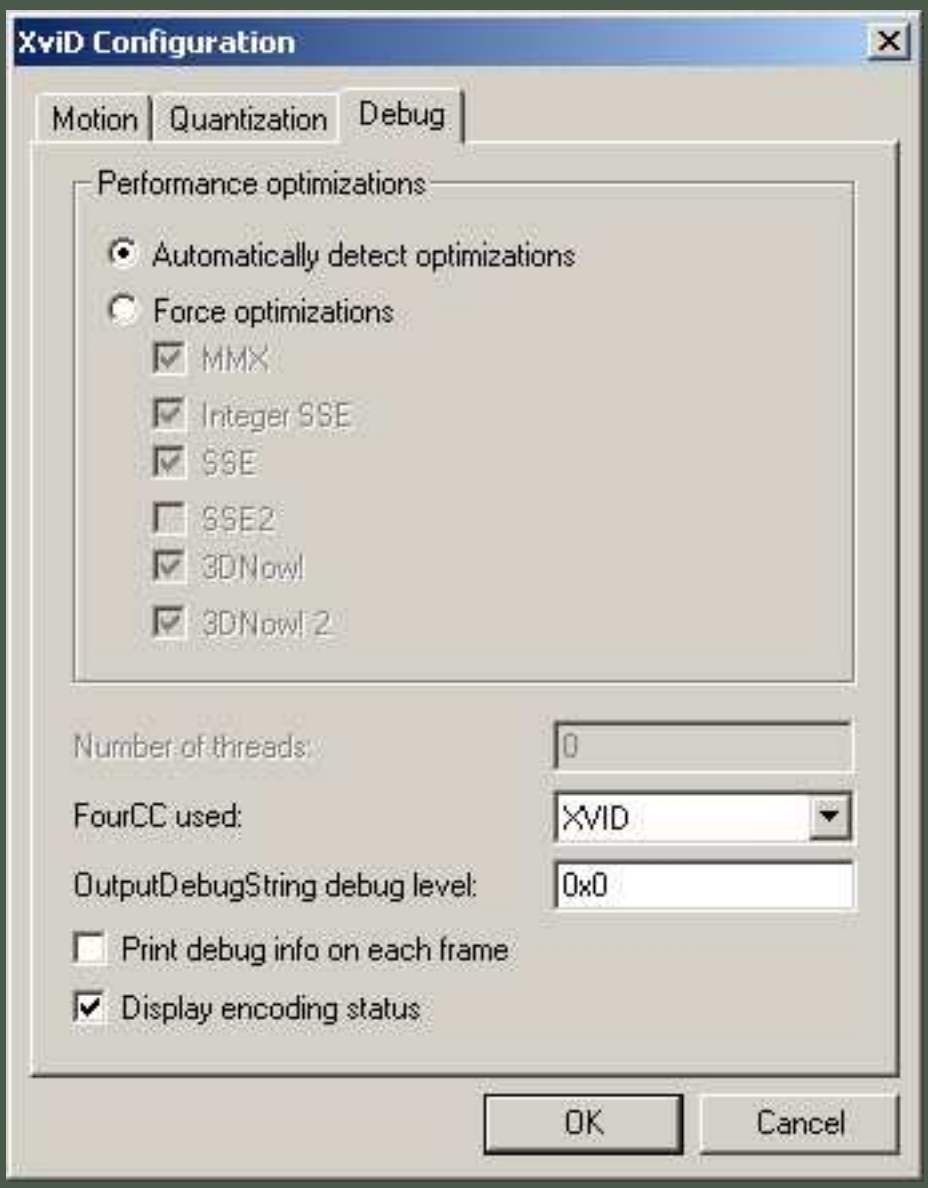
C13. 'Quantization' tab



Here you will find the settings for minimum and maximum Quantizer for I-, P- and B-frames. With all alpha builds they where set in the range 2-31 and setting the minimum to 1 would just reset it to 2. The reason for this is that Quantizer 1 really has little use and just inflates the filesize, with hardly any visible gain in quality.
With the Release Candidate builds the range has changed to 1-31 so that undersized files will become a thing of the past. Note: With RC1 quantizer 1 would break filesize prediction again (this time creating *oversized* files) but this has been fixed since RC2.
(Also, with older builds it was sometimes adviced to lower the maximum settings as well, to prevent frames getting lower quantizers than they really need and thereby creating visible artifacts. Currently I do not have enough information to say if this still holds up, but my guess is it's been fixed because I haven't heard much about it lately.)
Look here for a recent discussion about Quantizer use.

-Trellis
Trellis based R-D quantisation (Rate-Distortion quantisation) is more advanced and it works only with the quantization type H.263 in alpha builds. In beta and Release Candidate builds it also works with MPEG and MPEG-custom. It is a sort of intelligent second quantization pass, in which a more thorough DCT quantization examination is done by the Trellis process. In this process it can drop some coefficients (removing detail) and bring back other coefficients that were removed by the simpler standard quantization routine. Dropping coefficients hurts detail, but if bitrate savings are higher, it means that the codec will be able to use lower quants and retain higher quality.

C14. 'Debug' tab



The Debug tab holds some miscellaneous options that are meant for...hang on...debugging!

The 'Performance optimizations' part allows you to force optimizations for your CPU if it isn't correctly recognized by the codec. of course if you enable the wrong optimizations you will get, let's say, interesting results... Typically MMX should allways be on unless you use an old pentium 1, SSE and SSE2 is meant for newer Intel chips like the Pentium 2-4, and 3Now!(2) is for AMD chips.

CPUs and their instruction sets

	MMX	MMX etx.	SSE	SSE2	3DNOW!	3DNOW! Pro	AMD64
Intel Pentium	-	-	-	-	-	-	-
Intel Pentium MMX	Yes	-	-	-	-	-	-
Intel Pentium II	Yes	-	-	-	-	-	-
Intel Celeron	Yes	-	-	-	-	-	-
Intel Pentium III	Yes	Yes	Yes	-	-	-	-
Intel Celeron II	Yes	Yes	Yes	-	-	-	-
Intel Pentium IV, Celeron PIV	Yes	Yes	Yes	Yes	-	-	-
AMD K6	Yes	-	-	-	-	-	-
AMD K6-2/K6-3	Yes	-	-	-	Yes	-	-

Athlon, Duron	Yes	Yes	-	-	Yes	-	-
Athlon XP	Yes	Yes	Yes	-	Yes	Yes	-
Opteron, Athlon 64, Athlon FX	Yes	Yes	Yes	Yes	Yes	Yes	Yes

-'FourCC used:' - Here you can alter the FourCC used by the resulting file. FourCC is basically a content identifier code that is contained in the resulting video file. It tells the media application (like WMP, MPC or Virtualdub) what type of codec should be used to open the video correctly. You can set this for instance to DivX or DX50 to play with the DivX 5 codec. If you do, you have to take into account the limitations of that codec, so you can't use certain XviD features (like more than 1 B-frame or GMC). Not recommended really unless you want your files to play on a hardware player capable of DivX but not XviD.

The options 'OutputDebugString debug level:' and 'Print debug info on each frame' are really only useful for developers, so best to leave them alone. These will print all sorts of debug information on the resulting video.

-'Display encoding status' will popup an interesting window once you start encoding, showing you all sorts of statistical info like Quantizers used, types of frames used and amount of data encoded. Consider it a very fancy progress bar. If you don't like it you can turn it off here.

C15. Things that work but are not MPEG-4 compliant.

There are some things that you can quite easily do with XviD that are not *MPEG-4 compliant*. Not being MPEG-4 compliant means that it can quite possibly work with any build of XviD but is 'not meant to be' according to official MPEG-4 specifications. Now, since one major goal of the XviD project is to be fully MPEG-4 compliant this means that support for these non-compliant possibilities could be stopped/los/broken at any time never to return. So it's **higly recommended** that you refrain from using any of these possibilities if you want proper support for all your encoded material in the near and far future.

Currently known non-compliant situations are:

-Using non-mod16 resolutions: encoding clips to resolutions of which **both** width and height are not a multiple of 16 is considered a no-no and is not compliant. It is quite possible to do mod-4 resolutions with many builds (don't ask me how) but is not vey good for compressibility. The smallest MPEG-4 building block is not the 8x8 blocks but the 16x16 macroblocks, and using lower-than-mod16 resolutions will make the codec use macroblocks 'outside' the frame border....which is 'A Bad Thing'(tm).

-Using Modulated Quant or Modulated HQ. These were present in older builds and unfortunately many people did use them now and then, as they were new features back then and (as usual with new features) a request was made to test them. Basically the technique makes the codec switch Quantization Matrices from H.263 to MPEG and back whenever it suited it's compressibility goals (defined by the user).
Although the option has been removed a long time ago, clips encoded this way somehow still work though.

-The same goes for clips joined/pasted together in video editing tools like Vdub that use different QM's. A clip encoded with one QM joined with a clip encoded with another QM may work, but is not considered MPEG-4 compliant.

-Clips encoded with certain options that are joined with clips encoded with those options off can sometimes be non-compliant. A typical example of this is Qpel. I discovered a while ago that while Qpel usually doesn't help a lot in the main part of a movie, it can help a LOT in your average-day scrolling end credits. This unfortunately turned out to be non-compliant.

-The 'unrestricted' profile makes you build clips that are completely beyond any official MPEG-4 profile and gives you the full power of the XviD codec. It's quite obvious that you may break many official rules using this profile.

-Encoding beyond AS@Level5, especially in higher resolutions (like HDTV & SVGA) is not compliant to any MPEG-4 standard.

(As a sidenote: While many clip-joining stuff is not compliant, I consider it really weird that they should work at all in the first place. Perhaps someone more in the know could explain this. I wouldn't be surprised if some of these non-compliancies turned out to be unforeseen possibilities within the MPEG-4 specifications (especially the use of multiple QM's))

C16. Where do I find more documentation?

Iago's two pass walkthrough
get it here

VHQ Manual by Syskin
location

"The" Lumi Masking Thread
location

Snowbeach guide to Koepi's 24062003 build (pdf)
location 1
location 2

Snowbeach guide to Koepi's 14052003 build (doc)

location

Please note that many of these documents/topics are either for a specific build or rather outdated. They are still very informative though and it is recommended that you read them.

C17. I'm a newbie. What settings do you recommend?

Since this is beyond a doubt one of the most often asked questions, it certainly deserves an answer. The default settings will very likely get you good to excellent results on 'normal' source material like DVD's. If you like a more thorough answer than 'just stick with the defaults' you can follow some of the guidelines mentioned below.
However, they are not the 'ultimate' settings for everything as there's simply no such thing. Feel free to deviate (yes, become a deviant!) and experiment with the settings once you feel more confident using XviD. Using XviD always requires you to use your own judgement up to a certain point. The settings below are very safe and are here just to get you on your way.

-If something's not mentioned, don't mess with it.

-Always use two-pass encoding. Don't mess with settings between passes, keep them the same for both passes.

-Use Doom9's guides. They're especially targeted at newbies. Not reading them is A Bad Thing(tm).

-Only use target size, not target bitrate or quantizers.

-Profile: Use AS@Level5, nothing else.

-Motion Search Precision: use 6 - Ultra High

-VHQ: Use VHQ 1 on everything, possibly higher settings if you want 1 CD rips. VHQ 4 is perfectly safe, just slow.

-Quantization Type: use H.263 for a softer image, or MPEG for a sharper image (at the expense of bitrate, not recommended for a 1CD rip). Either way, use the same one for both passes.

-B-frames are good, but don't go above 2 Maximum B-frames (so put the number 2 in the Maximum B-frames box (putting in 1 will result in the same as DivX's Max number of B-frames)). Keep all other B-frame settings at default.

-Don't use Packed Bitstream.

-Don't mess with Interlaced encoding, greyscale, Debug settings, Curve compression, overflow treatment, GMC, Reduced Resolution, and Aspect Ratios. These settings are simply **not for newbies**; you need to understand them to use them properly.

-Don't use the Zones, they'll just complicate things for you.

-set 'Maximum I-frame interval' to 10 times the clip's framerate, nothing else.

-Qpel is safe but slow. Always use it if you get undersized files, it will increase quality.

-Chroma Optimizer/Motion is also good and recommended. Always use it.

-Use Gordian Knot to configure your rip and build your Avisynth files. Do **not** use Gordian Knot to do your encoding; Gordian Knot has no support for the new settings interface (yet; it's work in progress).
Also, do not use the compressibility test with Gordian Knot and XviD RC4; it doesn't work (properly).

-Do your encoding in Virtualdub or Virtualdubmod using an avisynth script. Don't use anything else, especially not VfApi. VfApi is Evil(tm).

-When Encoding with XviD make sure the encoding program uses Resize values that are divisible by 16. That means that the end result has to have both a width and a height that are divisible by 16. For instance 640x480 = (40 x 16) x (30 x 16). Using anything else is not for newbies and could be Evil(tm).

PART D: Troubleshooting

Back to top

D0. I get all sorts of garbage during play! (Also: DivX plays my XviD files!)

If you also installed the latest DivX, make sure you disable 'support generic mpeg-4' in DivX decoder configuration.
It overrides XviD as a decoder and can create all sorts of on-screen garbage if it's decoding XviD.
How to tell:
-DivX logo fades in and out at the start of playback...should be pretty obvious.
-Check your filters while playing...if it says 'DivX Decoder' it's also pretty obvious.

D1. I installed a new build but the settings don't work!

To quote Koepi:
hitting "load defaults" after first installation over an old alpha (dev-api3) -build is mandatory. Else you might screw up your encode.
Uninstall a previous version before you install the latest build, there have been a lot of changes recently and this is the best way to make sure you don't have problems with 'left-over settings' from old builds.

This is considered 'best practice' when using Koepi's builds.

of course, if you don't use any of Koepi's builds but others there might be other requirements. Hitting 'load defaults' is always a good thing if you update your codec.

D2. I get undersized files. What is wrong?

Nothing's wrong. An undersized file basically means that the codec sees no reason to make it any bigger. This is called 'codec saturation' and it is good news. If you want your file to be bigger, there are lots of ways to do so:
(all these ways take DVD-ripping as an example. If you have another use for it not all of these ways may apply)
-make sure all your XviD options have the correct settings
-make sure there are no errors in your filter settings and/or scripts in avisynth/virtualdub/whatever-tool-you-use
-Use a higher resolution. This simply gives the codec more information to put into the result.
-Switch to MPEG Quantization Matrix instead of H.263. MPEG QM preserves more detail and is sharper, but it requires more bits.
-Take out noise filters out of your avisynth script, **especially** those that are known to cause artifacts first.
-Reduce the strength of softening filters like warpsharp and unfilter. This can have a huge effect on filesize (and quality).
-Reduce the number of B-frames and try again. Also try more conservative B-frame settings.

In general, getting undersized files means you can take intermediate steps out of the encoding process that increase compressibility but hurt quality.

D3. I get oversized files. What is wrong?

-This is a bug in some builds when the filesize prediction/estimation process (also called 'Overflow treatment') is not properly tweaked.
With XviD RC1 the settings are not properly set to get a correct filesize when in the 'Advanced options' - Quantization tab the minimum quantizers are set to 1. Set this to 2 to re-enable proper filesize prediction. RC2 handles this better, but it still needs a bit of tweaking. Current version, RC4 needs testing feedback to see if it is fixed completely.
-Make sure you don't limit the quantizers too much. A maximum quantizer of 5-8 might give you oversized files.
-Also, regularly, users mistake "target size" and "target bitrate" - it's changeable, so make sure that the value you enter makes sense according to what the button shows.
-Messing with the overflow treatment settings is very likely to break filesize prediction.

D4. What to do if I get green and or pink blocks?

Some versions released after 2/2/2003 had this problem. It was fixed a long time ago so try out a newer build.

D5. I want to playback this movie I d/l'ed but there is no sound!

XviD is a video codec, thus has nothing to do with sound. Anyway, go to the download-section and fetch the audio-playback-filters for ac3, you're most likely lacking them. You may also need an Ogg Vorbis filter which can be found @ http://tobias.everwicked.com. Make sure you also get the subtitle filter from that site if you want subtitle-support. Nowadays AAC, another audio format, is also getting more popular.
Go here to download an AAC filter. The CoreAAC directshow filter is recommended.

D6. I only get a green/no picture on playback!

Several reasons are possible: your VGA card can't cope with the resolution of the video - install the DivX-G400-patch from the download-section.
You installed the Nimo pack? The Nimo Codec Pack is evil[tm]! Happy reformating and fresh-installing your system! After that, stick with ffdshow or the XviD decoders. Just install the codecs you need, and no bloated packs!
FFDShow can be downloaded here:
http://sourceforge.net/project/showfiles.php?group_id=53761

D7. I get choppy/stuttering playback!

This can be caused by a lot of different reasons, including the combined effect of two separate issues which form no problem individually. Here are some of the more obvious reasons:

-Underpowered PC. Make sure your computer has enough CPU power to properly play XviD files in general.
-Fragmented file. Make sure the file you're playing isn't heavily fragmented. Use a defrag tool on the drive where your file is located.
-Poorly muxed AC3-audio. Some tools (and fools) do a poor job when mixing audio and video together if the audio is AC3. Demux the audio and try playing the pure video file.
-Don't use crappy codec packs like the Nimo codec pack. Uninstall those and then reinstall XviD.
-Use a different player. Some issues happen only with certain players. Some players to try are: Media Player Classic, Zoomplayer, BSplayer, Windows Media Player 6.4 and last (and certainly least) WMP 9.
-When using ffdshow, try playing the file with different codec setting. Try playing it with XviD only, with ffdshow with XviD decoding, and with ffdshow doing the decoding. ffdshow can't handle clips with packed bitstream and more than 1

consecutive b-frame properly.
-Currently, Packed Bitstream has been known both to cause **and** to solve choppy playback, depending on all sorts of variables that are not completely understood. Mostly it has to do with certain incompatabilities between Packed Bitstream encoded clips and player software.
-More than 1 Consecutive B-frames may cause problems with some players. Especially ffdshow and the DivX 5.xx decoders.
-When encoding to the mp4 container format: mp4creator/mp4ui cant handle b-frames right when muxing to mp4 - don't use it! use the 3ivx mp4 muxer instead.
-Most hardware players using older chipsets can't handle more than 1 B-frame. Before buying a hardware player, do some research on it's capabilities.
-Sigma X-card owners: Drivers 1.3 of the x-card play back XviD with any number of consecutive b-frames and packed bitstream enabled. The newest drivers, version 2.1, produce stuttering in all XviD clips with b-frames, with any number of consecutive b-frames and with or without packed bitstream enabled. Get v1.3 of the drivers here.
More discussion here

D8. I get weird discolorations when playing back an old XviD file.

This can happen when you play back a file encoded with a build older than May/June 2003. Around that time XviD development switched a part of the compressing process to another type of technique, which is more MPEG-4 compliant. The technique is called DCT/IDCT and it was switched from 'simple' to 'walken'. The 'Simple' process encoded color information in another way than the 'Walken' process, and the latest decoders use the 'Walken' process to decode. Because of this discrepancy sometimes color information might be decoded differently, with discolorations as the result. The work on a solution to this issue is currently in progress. Since you can't detect the difference automatically, the decoder will have an option to switch decoding between 'Simple' and 'Walken'.

D9. I get no output on my TV-out!

This is **not** a problem with XviD. If your files play perfectly on your pc-monitor then it's not a codec fault at all.
This usually happens with not properly configured and/or installed graphicscard drivers, and less often with a fault in your operating system.
-When using windows, make sure you have at least DirectX 8 or higher installed.
-Do NOT use alpha or hacked drivers, like Omega's Ati or Nvidia drivers or similar. Use only WHQL-drivers if possible.
-Some videocards need an extra tool to properly configure the TV-out. 3dFX Tools for Voodoo 3-6 and Hydravision for Ati cards spring to mind. Make sure these are properly installed.
-Check the documentation of your videocard. You can also find help at several online sites.
Some examples:

Geforce FAQ website
Helpsites for 3DFX card owners
Ati (Radeon) support site

D10. I installed XviD on XP but it doesn't show up in the codec list!

This is a little known bug with Windows XP and has to do with how it handles codec registration in it's registry.
If you don't mind editing your registry open up regedit (Start --> Run --> Regedit) and go to:

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Drivers32]

You'll see a long list of codes with filenames next to them. The codes are FourCC codes. A FourCC code is stored inside a media file and is supposed to state the type of content. So a FourCC of 'Xvid' means it's an XviD file.
To the right of those FourCC codes you will see the codec files that Windows is to use to handle those types of content.
You will see a lot of empty entries, i.e. FourCC codes that aren't associated with any codec. Delete them all.
Yes, that's right: They can be deleted safely.
Apparently without these empty entries XP will show you all your codecs again.

There is also a tool called *BigFix* that fixes this and many other codec related bugs. Get it here.

(I wouldn't know: I think XP sucks ass and don't use it)

D11. My old XviD files don't play correctly with the latest XviD!

There are many probable causes for this, but some of the more obvious are:

-Resolutions different than mod16 (dividable by 16) were broken in old alpha builds. (If you read the guides back then you should have known to stick to mod16 or mod32 resolutions)

-You used interlacing with GMC and/or b-frames (also b0rked in alpha builds). Both Interlacing and GMC have only very recently been debugged.

-You used something else that was broken at the time.
-You did something way out of spec or not MPEG-4 compliant.

D12. Why do red areas in my clip look blocked or pixelated?

This is an issue with some decoding applications, it has nothing to do with encoding. The color space used by XviD is YV12 (check the glossary for more info) and is a very *lossy* color space. When decoding the decoder has to convert this to a full quality picture that the video card can send to either a monitor or a TV.
The lossy Xvid file has much less color information than uncompressed video, so the decoder has to reconstruct the lost information somehow if it wants to put out a proper result. Since there is way less color information in the Xvid than there is brightness information, the color information the decoder has to work with is far less precise. The underlying process is called ***upsampling*** and the color space gets converted from YV12 to RGB32, which isn't (very) lossy. Then the RGB32 signal gets thrown at you via your monitor or TV screen.

The problem occurs **when the upsampling isn't done properly**; Part of proper upsampling is taking that sparse color information and making a 'best guess' about what the values inbetween are supposed to be, creating a fine color gradient across the picture.
A process like this is usually called ***interpolation*** and if the decoder doesn't do that properly the decoded output will not have a proper color gradient but will look a bit blocky; After all the 'in between pixels' will not have an 'in between value' but the closest orginal value, which is far coarser, and creates very visible artifacts.
For some currently unknown reason the artifacts tend to be more visible in the red parts of the spectrum. Current best guess is that it is an interaction between YV12 color space-weirdness and the color-sensitivity of the human eye, unforeseen by the creators of MPEG.

Now the **bad** news is that this is an inherent issue with the color-space used by MPEG, not just with MPEG-4 but also with MPEG-1 and MPEG-2 (VCD and DVD). As long as the output was just on TV noboby seemed to care, but modern monitors and HDTV's are far more precise and it is now an obvious weakness in many MPEG applications.

The **good** news is that it's just a small issue with decoding and that it is very fixable. On windows these are the known fixes:

-First method is to use ffdshow with "ConvertToRGB32()" command in its AviSynth tab (use an ffdshow version that has it). This will make use of the upsampling features of Avisynth which work properly.

-Second method is to buy another video Card. Not recommended really...
Especially Geforce video Cards and older cards seem to suffer from improper upsampling methods. Trying another driver might help, but feedback on this has been too few and far between to really make any assumptions.

-A third method is to try to use the "VMR 9 Renderless" mode in MPC.
This will make MPC use another type of playback, possibly using proper upsampling.

-A fourth method (untested at this moment) that could hypothetically work is to first

open another media file and play this simultaneously with the file you really want to play.
(the idea behind this is that only one media playback can use 'hardware overlay' which makes use of the hardware acceleration features of your video card. The next file opened cannot use these features and would be played using software overlay, which might use other (and proper) upsampling.)
Needless to say this takes quite some extra CPU and it denies you the use of all the nifty hardware features your video card has. But, it might work.

D13. Other issues...

Asian versions of Windows do not show the Vfw interface for XviD properly.

Microsoft admits that it is a bug in windows(!). Though they don't offer any better fix than setting the locale to English, after that the dialogs are as expected. If you're one of the tweaking kind go to this forum thread for more information on interface adjustment.

Issues with Windows Media Player 9

WMP9 uses a new way of handling codecs, beside from a compatibility mode. But it turns out that (surprise surprise) WMP9 is still rather buggy. In fact, sometimes WMP9 opens up **two** instances of XviD to play the file, instead of one.
Best not to use WMP9 for now, it sucks anyway. What to use? ANYTHING BUT WMP9.

PART E: Feedback and Questions

Back to top

E1. I'm still lost. Who can I ask/call/email for support?

Well, *officially*, nobody really. Since there is **no single person, company or organisation developing XviD**, there is no 'support center', 'helpdesk' or something like that. Also, XviD is 'for educational purposes only' which means that *officially* it's not meant to be used seriously, if at all.
But, *in reality*, you will find plenty of people on the internet using XviD doing all sorts of imaginative and not-so-imaginative stuff with it, so there's bound to be somebody around to answer your questions. You just have to find them (which is difficult enough as it is, since the internet is **HUGE**).

E2. Where can I go on the internet to talk about XviD?

There are a couple of websites, mailing lists, internet forums and IRC channels where you can find more info, ask questions or talk with others about XviD. Some of these are better than others, and you may also find you like one type of communication more than the others.
Some XviD sites are www.XviD.org and www.Doom9.org. Both of these have forums where you can read all sorts of discussion and ask questions about XviD. The Doom9 XviD forum is highly recommended.
On IRC you can go to the #XviD channel on IRC server irc.freenode.net. Please note however that the IRC channel is usually more developer oriented and that you might be better off lurking in the forums a bit before entering chat.

E3. I want to report a bug. How do I do that?

Reporting bugs will help to solve issues in the code and incompatibilites with soft- or hardware. But first try to make sure it's not an issue with some other software or hardware component of your computer.
This means:

-Don't use experimental, unofficial or unstable drivers for your hardware. Use the latest non-beta drivers. If your OS is Windows, try to use WHLQ-drivers only.
-Don't overclock your system. Many crashes and bugs (even subtle ones) can be caused by components running out of spec.
-Certain hardware combinations are known to be buggy. Especially Creative soundcards on motherboards with Via chipsets create a lot of problems.
-Make sure your operating system is stable and has the latest patches. We're not talking about security-patches but patches that affect stability.
-With Windows, make sure your desktop color depth is 32bit. 16bit color depth will make the video look artificial.
-If you have an issue with encoding, make sure you have everything setup properly before reporting a bug.
(Try to eliminate any faults in other steps in the proces, like a buggy Virtualdub version or an error in your avisynth script)
-If you have an issue decoding (playing) a file, switch decoders and see if you can isolate the problem to a single decoder. Make sure you're playing Xvid files with the proper decoder. DivX now automatically sets an option to play 'generic MPEG-4 content' so disable that one.
-If you're using an untested alpha build, like one of Umaniac's instabuilds, you have to face the fact that you're pretty much on your own. **These builds are not likely to be used by a lot of people and are completely unsupported, so your chances of finding anyone to confirm your bug are small.**
(Sticking with major builds has the advantage of having other people around that can confirm bugs)

If you have ruled out all of the above and are still convinced you have a bug to report, go to Doom9's XviD forum and submit a post containing your bug report. Please

follow the following guidelines when posting:

-What version & date of XviD are you using? (i.e. Nic's, Koepi's, uManiac's, etc)
-In Windows, what DShow filter are you using to decode it? (i.e. Nic's, Koepi's sysKin's, ffdshow etc.)
-Does the bug occur with every encode you do or just the one clip? Big clip or small clip?
-Your computer config. (i.e. Operating System, Media Players, Hardware (specifically: graphics card, Operating System, CPU))
-Please post all the options you used in your encode (i.e. 2pass, GMC, Q-Pel, Bitrate, VHQ, etc)
-If using AviSynth, post a summary of your script and check that it plays and looks fine in VirtualDub before posting.
-On a forum, try to fit a very brief summary of your problem into the subject line (rather than just having a subject of "help!" or similar, that's likely to get you kicked off that forum for being too unspecific)
-Please specify the container format you are encoding to/playing from.
-Specify if your encode is muxed with audio (specify the audio) or not.
-make sure the bug is not reported before. The usual place for this is at the XviD forum at http://forum.Doom9.org.
-Report a bug only to the specific build-topic. Major builds are announced in build-topics.
These are also the first place to look for a list of known bugs.
-It is always beneficial to confirm a known bug. This might rule out certain assumptions, like bugs occuring only with an AMD CPU if you have a Pentium IV. Also if someone reports a bug on a setup just like yours and you don't have that same bug, report it. Keep a confirmation short and functional.

E4. How can I contribute to the XviD project?

So you want to help out, eh? Well, help is always appreciated, and while XviD is already one of the finest and most highly-developed MPEG-4 codecs out there, it's still lacking finesse in certain respects (most notably at this moment documentation and code optimizations) and there is always something to do to help if you want to.

In descending order of effect, these are *some* of the things you can do to help the XviD project:

-Hire someone to develop XviD fulltime
If you've got the cash (say you're a company doing media development or producing XviD support for embedded devices) you could hire one of the XviD developers fulltime or parttime to do development, programming and testing on XviD. This would really speed up development as most of the developers are either still in school or university or have fulltime jobs, and they develop XviD in their spare time. Putting one (or more) of them on a paid job developing XviD would mean there's at least a guaranteed level of continuous development.

-Give all/some/one of the developers a new pc
If you don't have the dough to put somebody on a payroll but still want to contribute with money you could donate hardware to the developing community. This would allow them to do much more work since most of them use their own computer at home for all the programming, compiling and testing. This also means that XviD testing is limited to a small amount of software and/or hardware combinations, mostly focused on mainstream low-end and mainstream computers meant for home use. So for example if you want multi-threaded support of support of Athlon64 or Xeon optimisations, or you want it to work on a supercomputer or Beowulf computer, you'd have to give a developer at least **access** to these type of systems. I'm not saying that it would be a good idea to give Koepi or Nic or one developers a Cray supercomputer (would be nice though ;^)) but giving them access to one so they can program support for it would certainly be a good way to help further development. of course if you're a BIG hardware company like Intel or AMD, it really helps if you give the developers access to the newest and latest hardware you're developing, or give them versions of the newest hardware optimized compilers like ICL and similar.
If you're not that loaded with cash, but want to help out in a smaller way, you could of course always give them that spare old Pentium III or that stick of memory you have sitting in your desk collecting dust. Any small contribution helps.

-Become a developer
Let's say you don't have money but you've got brains, a computer and some fine programming skillZ(tm)...
You could become a developer yourself and start collecting bugs, builds, headaches, cafeine intoxications, fame and glory just like Koepi, Nic and the others! Becoming a developer is easier than you might think. XviD is an open source project, so everybody can download the source code, look at it, alter it, test it, alter it again, test it again, alter it again, test it again.... straight into eternity. Any alterations you make that are considered better than the previous code will contribute to the further development and glory of XviD. And of course you will get listed here in the appendix as one of the developers of XviD ;D.

-Help with documentation (like this faq)
Although nowadays there is more documentation on XviD than there ever was before, there is always room for improvement. (If we ever get to the point where there's so much documentation that we have to start selecting, it will be a good day indeed.)
Typical examples of documentation are, in no particular order:
= Specific build guides, (like Snowbeach guides)
= Elaborate and detailed explanations of the workings of a particular option (like Qpel, B-frames, Adaptive Quantization etc,)
= List of known bugs for different builds and workarounds
= More general documentation using XviD in a broader perspective (like a Gordian Knot or AutoGK guide using XviD)
= FAQ's (like this one)
= Good, readable MPEG-4 documentation (or a list of things that are and aren't MPEG-4 compliant)

-test XviD on all sorts of different software and hardware combinations
You can always help out the developers by testing their newest code for them. You can either download the latest source and compile it for yourself, or you can use one of the builds supplied by the devs.
Testing it means you take a videoclip and you encode it and try to play it on all sorts of combinations of software and/or hardware, and then report the results. A proper bug report (see question E3 for how to file a proper bug report) can really help out in finding and solving problems especially ones that are hard to reproduce.

-Help people with XviD on fora, irc and irl
You can always try to help out other people who are trying to use XviD but don't understand certain things about it. Usually the devs are rather busy doing all sorts of difficult programming stuff and they really don't have the time to answer all the questions that people are asking all over the place. So, it doesn't hurt if you take some load off their backs by asnwering questions that you have seen asked and answered many times before. And even if you don't know what certain options do, or have no idea how the internals of XviD work, how to compile it or test it, if you know someone who does or a website or internet forum where answers can be found, chances are you have already helped that person a lot.

-tell your friends about how cool XviD is
Spread the word, man, spread the word....the bigger the community gets the more people are bound to contribute to the project in one way or another. It's simply a matter of numbers...the more people know about XviD, the more people will download it, use it and test it, get involved in the community, give feedback, become developers, help other people out solving bugs and answering questions...
All it really takes is a big mouth and some power of persuasion, and if you haven't got brains, money, writing ability or the time and CPU cycles to test XviD, you can always try to persuade people that do. ;^)

NOTE: If you have illegally downloaded a copyrighted movie then you will get no other help here! We do not help with piracy of movies!

Appendix

Back to top

A. Useful Links

XviD main development site: www.XviD.org
Doom9's audio/Video editing site: www.Doom9.org
Doom9's forum
XviD Sourceforge site: http://sourceforge.net/projects/XviD/
Koepi's Media Development Homepage
Nic's XviD page
Sourceforge site of Media Player Classic and Matroska
Ffdshow homepage
WinCVS download site
Debugview site
A introduction to interlacing and deinterlacing.
The DVD FAQ. Very good reading with lots of info.
Multimedia Data explained.
Must read for DVD2Avi users:
DVD2AVI FAQ
Force film, IVTC, and Deinterlacing
Technical documentation concerning An introduction into TV standards
Info about color spaces.
An article on the chroma upsampling error.
A document outlaying the differences between pixel sizes: Square and Non-Square Pixels
A document outlaying the differences between video AR standards: A Quick Guide to Digital Video Resolution and Aspect Ratio Conversions
Rarewares, the source for many hard-to-find audio and video files.

B. Downloads

Ogm directshow filter. This will add directshow support for the playback of Ogm files.
Primary location Alternate location

Ogm directshow subtitle filter. Use this to get subtitles working with Ogm files.
Primary location Alternate location

Koepi's XviD-1.0-RC4 build. Latest build as of 6 april 2004. If you want to use one of Koepi's builds, use this one.
Primary location is Koepi's homepage. Alternate location

Koepi's XviD-1.0-RC3 build.
Alternate location

Koepi's XviD-1.0-RC2 build.
Alternate location

Older builds:
Koepi's XviD-1.0-RC1 build. Alternate location

Koepi's XviD-1.0-beta3 build. Alternate location

Koepi's dev-api-3 build 24-06-2003. This is a rather stable alpha-build. Alternate location

Koepi's XviD-1.0-RC1 Decoder. RC1 Decoder only. Please note that there are newer versions around that solve some issues.
Primary location is Koepi's homepage.Alternate location

Media Player Classic 6.4.6.7 for Windows 2000, Xp and 2003.
Primary location is it's sourceforge homepage.Alternate location

Media Player Classic 6.4.6.7 for Windows 98 and Millenium.
Primary location is it's sourceforge homepage.Alternate location

Ffdshow build of 24-04-2003.
Primary location is it's homepage.Alternate location

Ac3 audio directshow filter version 0.70b. Use this for playing back AC3.
Primary location is it's sourceforge homepage.Alternate location

C. Glossary

3ivX: A relatively new MPEG-4 implementation very similar (but still inferior) to XviD and DivX. Like DivX it's a commercial MPEG-4 implementation. It's available for many platforms, including Windows, Linux, MacOS, Amiga and BeOS. It's website is here.

ABR: **A**verage **B**it **R**ate. Encoding in ABR means that the bitrate can vary (wildly) per measured unit of time (either seconds, milliseconds or other standard; for XviD it's per frame), while still keeping the whole encoded result to a mean average value. ABR is a term most commonly used in audio compression, and many audio compression formats can encode using some form of ABR. However, usually it's not the most optimized form of encoding and especially mp3 and Ogg Vorbis deliver better results when encoding in VBR-mode.
For XviD the situation is different though. In two-pass mode XviD encodes to a set size, which could be interpreted as another way of ABR (a set size for a set length). In sharp contrast to audio compression this mode is highly optimized and gives by far the best results.

CBR: **C**onstant **B**it **R**ate. Encoding in CBR will give each time-based unit exactly the

same amount of bits, without any regard for the complexity of the signal that is encoded. This is one of the oldest forms of encoding, especially in the audio compression world, and is the least optimal (as it gives complex signals too few and simple signals too many bits).

DivX: A MPEG-4 codec originated out of an illegal hack of a Microsoft codec, which became widely known as Divx 3.11. Later a new development started, designed to remove all proprietary code, with the DivX 4 (also called OpenDivX) codec as a result. The latest DivX versions (current one is 5.1.1) are commercial MPEG-4 codecs, with a less advanced implementation still available for free.
While DivX is considered to have better hardware support it's quality is considered (slightly) inferior to XviD, at least by XviD users and developers. :^) Go take a look at their website.

FourCC: "Four-Character Codes (FOURCC), a set of codes that are four characters in length, was introduced by Microsoft to clearly identify video data stream formats. The unique FOURCC value assigned to every compression format and pixel layout allows video frames to be passed between file and codec by ensuring the FOURCC of the source frame matches a FOURCC supported by the codec." Read the rest here

Interpolation: The process of finding a value in between two other values by taking the average of those two values. If you take a white pixel and a black pixel and want to create a third pixel in between by interpolation, it will end up being grey.

MPC: Short for **M**edia **P**layer **C**lassic. It's a Media Player with an interface very similar to the Windows Media Player 6.4, which comes with Windows 98 and 2000. But it's much better and there are a gazillion differences under the hood. It's considered better than **any** media player Microsoft has. MUCH better.
Go look for it over here

MPEG: **M**otion **P**icture **E**xpert **G**roup. A conglomerate of companies and institutions that sit around a table once in a while making up new standards. Creators of the MPEG-1 (VCD), MPEG Audio Layer-II (mp2), MPEG Audio Layer-III (mp3), MPEG-2 (DVD, SVCD), MPEG-4 (3ivX, DivX, XviD, etc)) and other standards, they are a very important factor in the media industry.

Ogg Vorbis: Container and Audio format combination. It's an often incorrectly stated term for the use of the Vorbis audio format. Officialy, Ogg is the container format in which audio can be placed. Vorbis is the audio format developed by the designers of Ogg in conjunction with it, and Vorbis can only be placed inside an Ogg container (AFAIK). Therefore it's called Ogg Vorbis.(The Ogg container can also hold other formats)
As for the Vorbis audio format, it's a new, open sourced audio format meant to surpass mp3 in both quality and compression, and it succeeds in this. In sharp contrast to mp3, which is mainly **CBR**-based with VBR sort-of developed on top of this , Vorbis is VBR-optimized from start, and CBR and ABR modes are considered sub-optimal (but still better than mp3).

If you want to use Ogg Vorbis with your XviD files, you can't use it with the avi-container. Use either Ogm, MP4 or MKV.
Official Vorbis Website is here.

Ogm: **Ogg** **M**edia container format. An unofficial standard developed by a guy called Tobias Waldvogel primarily to make possible the use of Ogg Vorbis audio with video contained in Avi files. The avi file and the Ogg Vorbis file are bound together by a muxing tool into a Ogm file. It has been developed further to include support for multiple audio files, subtitles and more.
To play this type of file you need a splitter filter to demux the Ogm. You can get one for win32 here.

RGB32: A type of color space that uses almost no compression. The primary color values **R**ed, **G**reen and **B**lue are stored in an 8-bits value on a per-pixel basis. This means that each pixel has three 8-bit values totalling 24 bits, with one zero byte making a total of **32** bits per pixels (The last byte is added because it makes it easier to process this type of video for 32-bits processors). Needless to say, files using this color space can get quite big.

VBR: **V**ariable **Bi**t **R**ate. Encoding with VBR means that the bitrate can be different for every time-based unit of source. Basically a time-based unit means that bitrate can differ per standard of time (hour, second, millisecond or the time it takes for you to take a leak), but the resolution of XviD is not in seconds but in frames per second. (VBR is more an audio compression related term anyway)

Video Renderer: Part of the DirectX interface responsible for outputting decoded video. Irritatingly, both VMR and this Video Renderer call themselves 'Video Renderer' towards the user. Look here for more information.
More general info can be found here.

VMR-7/9: VMR stands for **V**ideo **M**ixing **R**enderer and is a part of Microsofts DirectX implementation. VMR-7 is only available on Windows XP; VMR-9 is installed as part of DirectX 9 and works on Windows 98(SE), ME, 2000, XP, and 2003. It's a new video processing interface but still not well supported. Since VMR-7 is only available on XP, expect even less support. VMR-9 is **not used by default** on systems with DirectX 9 installed, the default is VMR-7 for XP and the older (but more compatible) 'Video Renderer' on 98(SE), ME and 2000.
For more (but very technical) information look at the MS page for VMR-7 and VMR-9

YV12: The color space used by XviD. A Luminance value is stored for every pixel, but for only every four pixels both a U and V value is stored. Four pixels then have 4x8 +8+8= 48bits stored. 48/4 equals 12 bits, so this color space takes up about **12** bits per pixel.

D. XviD Developers

E. FAQ Contributors

In alphabetical order:
Crusty
Doom9
Koepi
mikeX
Nic
Snowbeach
SysKin

Changelog:

6 april
Updated FAQ for RC4; added Meta-tags for description.
7 march
Experimentally added webcounter
4 march
Yeah...pictures! Also alterered some stuff here and there, mostly as a sideeffect of adding pictures.
1 march
Updated for Release Candidate 3; Minor alterations here and there; Updated PDF version(thanks to atreya2011).
25 February
Added stuff to glossary; minor alterations to A7, B1, B4, and C17
21 February
Minor edits to B-VOP sensitivity and Turbo
20 February
Added C17; updated D12; added stuff to glossary; some minor alterations
19 February
Update Qpel and cartoon mode; Added Glossary to appendix; alterations to RRV; minor alterations/corrections across the field.
16 February:
Many typos and style corrections all over the place; Added D10 D11 and D12; Small alterations here and there, nothing fancy.
11-14 February:
Many typos and style corrections all over the place; added B8; added C3a and moved the rest up; filled in C9; added C15; added D8;
Major content alterations to B7 and B6; Added timezone to last-update-time;
Minor content alterations to A4, B4, C1, C3f, C4, C5, C8, C11, C16, D1, D7 and Appendix.
10 February:
A4 complete, waiting for feedback for error correction
B4 added note about instabuilds. C3c: Filled in C5: 'More' Profile options - Aspect Ratio tab Filled in. Recommended it's use 'for advanced users only'
C14 'Debug' tab Now filled in. D1: Added note about RC2 having new installer
Moved D8 to D9 'I get no output on my TV-out!' and added D8:'I get weird discolorations when playing back an old XviD file.' (About the IDCT simple/walken issue...AFAIK the only effect is discolorations)
E2. Where can I go on the internet to talk about XviD? Filled in
E3 bug reporting: Added stuff from Nic's sticky on the forum and additional stuff. E4 contribute: filled in completely now
Added lots of stuff to the 'useful links' section. All across: minor error and spelling corrections
Last week of January 2004:
Start of new XviD FAQ project. Doom9's FAQ was the starting point. Now completely rewritten.