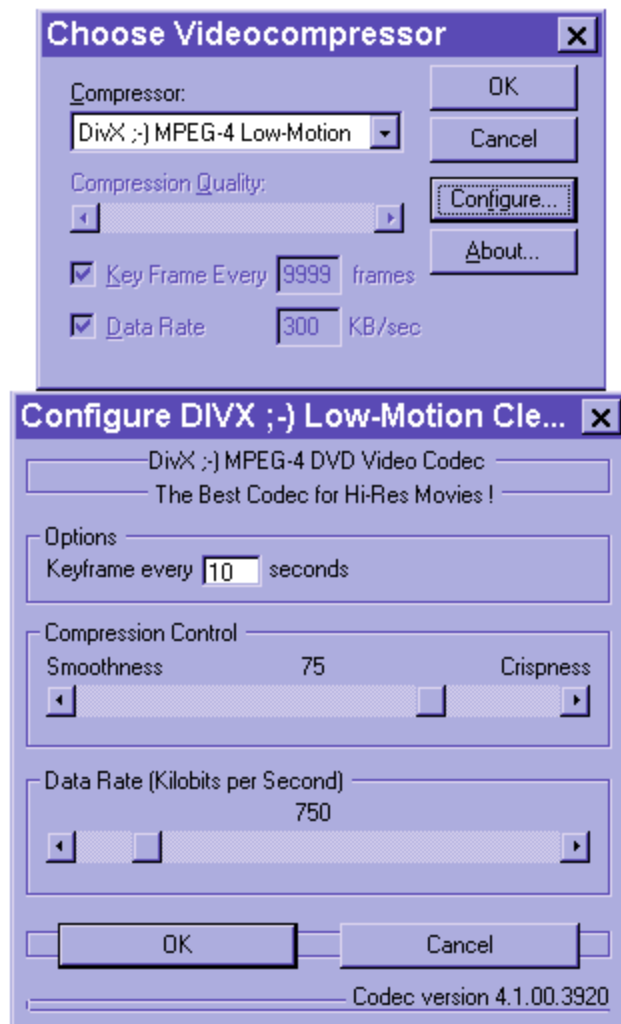


DivX Quality & Bitrate Guide

Everyone is looking for the ultimate settings! The ones that will make their DivX movie have the very best possible picture quality for the smallest possible file size! I too have spent much time searching this dark road. And now I feel its about time I gave everyone a few pointers from my experience =). These are just a handful of tips that I have made and should not be taken as though I am to telling you what settings you should be using. Rather, I hope to save you some time and help you to make good choices for yourselves.

INSIDE THE DIVX CODEC

Before we can decide on the best setting to use for our movies, it would be good idea to start by looking at what all those settings on the DivX codec do and what the differences are between the Fast Motion and Low Motion codecs.



KEYFRAMES

Most video formats that use compression will have keyframes. These help the video player to seek to various parts of the movie easily and to keep the picture quality good. They also take up the most amount of memory of any video frame. Most codecs will use one keyframe between every 5-10 seconds. The default setting on the DivX codec is 1 keyframe every 10 seconds. This is fine most of the time and changing it will not really increase the image quality enough to notice. In the past people started making DivX's with a keyframe every 9999 seconds to keep down the file size. The result was that you couldn't resume watching the video from where you left off! Every time you played it you had to watch it from the start again!

BITRATE

There are two kinds of bitrate setting used on video compression. Constant Bitrate (CBR) and Variable Bitrate (VBR).

Constant Bitrate: constant bitrate represents how much memory per second a codec will use to encode a movie. Obviously, the higher the bitrate the better the picture quality. Say, for example, one second of video was 25 frames. If I set a bitrate of 1000 Kilobits per Second (kbps) and encode 1 second, each frame would use 40 Kilobits of memory each. If I used a bitrate of 2000 kbps each picture could use 80 Kilobits. Obviously an 80 kilobit frame will look much better than a 40 kilobit frame.

Notice too that it doesn't matter what resolutions we use! Lets say we encoded one second of video at 1000 kbps at a resolution of 352 x 288 pixels - the final filesize would be exactly 1000 kilobytes! Again, if we encoded one second of video at 1000 kbps, but this time we use a cinema resolution of 5000 x 5000 pixels; the final movie clip would *still* only be 1000 kilobytes in size! I know it sounds strange, but we set only '1000 kilobits' for one second of video and that is *all* it will use!

Variable Bitrate: a variable bitrate codec sets its own bitrate in accordance with the movie. To illustrate, mpeg compression works basically by saving only the differences between sequences of frames. In non-action scenes there are very few differences between frames. Hence, if the total amount of differences are "small" the kilobits needed to store them will also be small. On the other hand, if there are lots of differences between frames it will take "lots" of kilobits to store them.

It is impossible to know how much memory a Variable Bitrate encoded movie will take because it all depends on the amount of Fast Motion and Low Motion scenes it has. Nevertheless, most variable bitrates codec have limits. For example, DVD's are often encoded with a certain maximum and minimum bitrates that will make sure they don't use too much memory and go over the target filesize or use so little memory that it gives a bad looking picture.

SMOOTHNESS & CRISPINESS SETTINGS

One trick people use on AVI files to reduce the filesize is to use a smaller framerate. For example, NTSC movies use a framerate of almost 30 frames per second (fps). If we only used 15 fps then we have effectively halved the filesize! This doesn't look too bad either especially on cartoons which are usually only produced at 15fps anyway!

But although this worked for uncompressed AVI's, no Mpeg format could do this! Say, for example, we tried to encode an Mpeg-1 file at 15 fps. The mpeg codec would automatically add extra frames to make the video at least 23.976 fps which is the speed captured with a motion picture camera! Then it would add a code at the start of the movie to tell the video player to only play it back at 15fps! In other words, the movie would still take the same amount of space as a 23.976 movie but play back at 15 frames a second! When I first wrote my previous quality guide I thought this rule must also applied to DivX, but this was not quite correct. The DivX codec is able to "drop" frames in order to save space!

And here is where the DivX smoothness and crispiness settings come in. The smoothness and crispiness settings refers to framerate preservation. The smoother you set it the less frames are dropped. If we set the crispiness high then it will tend to drop frames in order to save space, but the playback will start to become more jerky! But before you start panicing, this frame dropping effect only happens at very very low bitrates. At the bitrates we would normally encode a DivX, dropped frames are not a big problem. Nevertheless, while I used to say just use 100% crispiness for best quality, now I think keeping it to about 75 is safer. My tests have convinced me that you should not suffer from any dropped frames or jerky playback this way.

Framerates and Bitrates: Finally this brings me to a strange fact. If we encode a DivX at 30fps it will actually end up smaller than the same DivX encoded at 25fps! I cannot really explain this, but I have a theory. A 25fps movie encoded at 1000 kbps would (at a constant birate) use 40 kilobits per frame. But a 30fps movie encoded at 1000 kbps will use about 33 kilobits per frame. 30fps movies are basically 24fps movies with "repeat" frames added to take up space. Now, since, for the most part, Mpeg-4 only records the differences between frames the total differences between one movie of 25fps and the same movie at 30fps is zero! But the codec is now still only using a bitrate of 33 per frame. This is possibly what makes the smaller filesize, which in turn, gives a lower picture quality.

WHATS THE DIFFERENCE?!

"Hi Folks !

Always the same question, always the same answer... The LowMotion codec is a hack from version 4.1.00.4920 of the M\$ MPEG4v3, the HighMotion codec is a hack from version 4.1.4917 of the M\$ MPEGv3.

I really don't know the internal difference between the low-motion and the high-motion. I NOTICE the difference and decide to make two version, the low-motion which came from the beta version and the high-motion witch is the version 4.1.00.3917, the newer builds looks like be an low-motion style... The low-motion and the high-motion are the same file but with differents builds ! As you know I'm not the coder of this thing so I could not help you more... I mainly use the low-motion which produce better half-toning results, and the high-motion when the picture is very moving.

The final bitrate is extremly related to the CONTENTS of the video, a fast moving, very detailled picture is harder to code than a almost still, very clean picture... I think that the requested bitrate is a MAXIMUM bitrate, and the differences are in the way the encoder try to match this bitrate, the low-motion seems to allocate more bit to color and is 'tighter' to the requested bitrate, the high-motion codec allocate more bit to the luminance but is not as 'tight' as the low-motion is.

Gej

<http://divX.ctw.cc>"

As you probably guessed the above quote is from Gej the person who made the DivX codec! All Mpeg-4 codec's are just hacks of the Micro\$oft ASF codec. This includes SmR (nAVI) and even the Angel Potion codec (according to Avery Lee) which claims to be the first one that isn't a hack. To be fair the Angel Potion does seem to perform a little differently from the rest and is perhaps a highly altered ASF. But either way, at the time of writing this article it was filled with bugs and so is not, in my opinion, a good choice to use. When I say they are "hacks" the only real differences are they allow just about any program to use the codec to encode Mpeg-4 files. There was no alteration to the codec at all - its just like making a hack for a time limited trial, the hacked program itself remains unchanged. So there is no real difference in quality between various Mpeg-4 hacks - all look basically the same quality!

This brings us to the big question of which is better, ASF or DivX! And why bother to hack it anyway? Well, Micro\$oft decided no one could use Mpeg-4 unless it was encoded with their crappy encoder. ASF files were only permitted to be produced at very small resolutions. ASF's when compared to pure Mpeg-4 files would add almost 100MB's of extra data to the average 650MB movie! This was due to bulky overheads and all the stuff designed to make them stream over the internet. ASF's didn't allow MP3 audio and almost always ended up with serious audio synchronizatioin problems. Later versions of Media Encoder only allowed usage of the Fast Motion codec which produced bad image quality for low motion scenes! In short, DivX solved all these problems and anyone who says DivX is just a joke and ASF's look better just doesnt know what they are talking about!

FAST MOTION VS LOW MOTON CODECS

Gej was correct in his observations of the Fast and Low Motion codecs. BOTH codecs are variable bitrate which means the final filesize are very hard to predict.

Fast Motion Codec

The Fast Motion codec is the hardest codec to predict a final filesize. You could encode one movie at 6000 kbps and another at 900 kbps with the Fast Motion codec and still end up with two movies of almost the same size! Or the one at 6000 kbps could end up double the size of the 900 kbps one! There is just no way to tell! When we set the Fast Motion's bitrate we are setting the *Maximum* bitrate! This means it will always use an average bitrate of about 300 kbps until it reaches a high action scene and only then will it increase the bitrate to the maximum level! This is why people get confused, because it doesn't use the bitrate they put in until it finds an action scene!

Low Motion Codec

The Low Motion codec, on the other hand, uses a *Minimum* bitrate. This means that it will hardly ever go higher than what we set it. So if we set it to 800 kbps it would use 800 kbps on most scenes and only use a very little more or less depending on the action. This means the Low Motion codec is a lot more predictable.

COMPARING THE TWO CODECS

The following examples will make clear the strengths and weaknesses of the two codec's.

Fast Motion codec

Here is a high-action scene compressed with the Fast Motion codec at 6000kbps:



Here is a non-action scene compressed with the Fast Motion codec at 6000kbps:



Low Motion codec

The following is a non-action scene encoded with the Low Motion codec at 600 kbps:



Here is a high-action scene compressed with the Low Motion codec at 600kbps:



From the above examples it is clear that the Low Motion codec always looks better on non-action scenes even if it uses a bitrate as low as 600 kbps. It is noteworthy too, though, that when the Low Motion codec goes below 600 kbps it will not look significantly better than the Fast Motion codec unless the Fast Motion codec is set to the same bitrate too.

The Fast Motion codec also has an upper limit of about 2000 kbps. It doesn't seem to make much noticeable difference to quality whether we choose Fast Motion at 2000kbps or 6000kbps! It does, however, make significant demands on filesize.

Finally, as we can see from the examples below, when the Low Motion codec is set to between 1000-1500 kbps it starts to look overall better quality than the Fast Motion codec no matter what we set it to!



Fast Motion 6000 kbps



Fast Motion 2000 kbps



Low Motion 1500 kbps

CONCLUSION

The Fast Motion codec saves the most space and does very well on high action scenes. If you are going to use it for movies always set it to 2000 kbps. There is little point using any other setting and it has the best quality to size ratio. You might consider using the Fast Motion codec for putting long movies on a single CD.

The Low Motion codec set at 600 kbps will do better job than the Fast Motion codec on just about all low action scenes. It cannot compete with the Fast Motion codec on action scenes until we set a bitrate of between 1000-1500kbps. After which, it starts to produce much better results than the Fast Motion codec.

Final Note: At very small resolutions such as 320 x 240 the Fast Motion codec does a very bad job. It is probably better to use the Low Motion codec all the time in such cases. But, as always, do some testing and see what you think.

QUALITY AND RESOLUTION

All Mpeg formats, including DivX Mpeg-4, break up the picture into 16 x 16 blocks called macroblocks. Each block is allocated a certain amount of memory based on the bitrate we enter into the codec. The more the bitrate the better the quality. Lets see what happens to the macroblocks when we allocate less and less bitrate to them. The best way to see this is to enlarge the video so we can see better, like this:



Take a look at the images below. The first one was given the highest bitrate and the lowest was given the least bitrate. As you can see, the less bitrate we give it the more simplified the macrblocks get!



Ideally we would love to encode all movies at 720 x 576 full DVD resolutions. But at this resolution the image is broken up into 1,620 macroblocks. It has an effective resolution of 45 x 36 blocks.

Lets say we have decided that to fit a movie on a single CD as DivX we needed to use a bitrate of 800 kbps. But when we look at the picture at 640 x 480 we can still see many of these macroblock artifacts. We cannot increase the bitrate or it will not fit. Many think if we make the image big (i.e. full sized) the blocks will be smaller. This is wrong! Larger images means more blocks which means less memory is given to each block. When each block has less memory allocated to it, it becomes simpler and simpler until in the end most of the blocks become nothing more than blank colours!

The only solution is to make the image smaller. Smaller images means less blocks which means more memory is allocated to each block and each block looks more like the original picture. A resolution of 480 x 384 would give us 30 x 24 macroblocks, which is 900 less macroblocks. This can make a big difference in quality!

The above methodology is nothing new to DivX, the same has always applied to all Mpeg formats since they were designed. Because of the way it is compressed the picture resolution is not the key factor in determining the image quality. Smaller pictures actually look better than larger ones at the same bitrate. The technique for making the best quality image is simple mathematics:

1. Use the highest bitrate you can.
2. If the image shows too many macroblocks shrink the resolution a little.
3. Check it again. If there are still too many macroblocks shrink it a little more until they are not noticable. Its unlikely you will get rid of all macroblocks on a single CD DivX but you can get rid of most of them.

Commercial Video CD's (VCD) are considered by experts to be almost VHS quality video. Yet they only use a resolution of 352 x 240! This *is* possible even though a TV resolution is something like 576 lines. It is hard to compare an analog image to a digital image by mere resolution. For one thing VHS uses signal compression, yes, that's right, VHS is also a compressed format! But the response curve of VHS places -3 dB at around 2 MHz of analog luminance bandwidth is equivalent to 200 samples / line. VHS chroma is considerably less dense in the horizontal direction than MPEG source video. And from a sampling density perspective, VHS is superior only in the vertical direction, but when taking into account interfield magnetic tape crosstalk and the TV monitor Kell factor, not by all that much. Well, that what I read anyway! But the conclusion of the matter is using small resolutions than TV lines can still produce video's almost as good as VHS.

I have only a couple of rules I always follow for deciding video resolutions. I never make it larger than 640 x 480 (which is higher than TV resolutions) and I never go lower than 240 pixels high unless absolutely nessasery.

CROPPING THE VIDEO

Considering the previous facts about macroblocks, it makes sense that cropping out the black bars found at the top, bottom and sometimes at the sides of a movie would allow the codec to allocate more memory to image quality. But the amount of memory allocated to a pitch black area is quite negligible. So the most vital thing to remember for optimal compression is to always crop a few pixels into the image so as to delete all of the black bars. Mpeg compression works best on blurry images; so if a hard black line is seen at the edge of your cropped movie it will not compress as effectively. In fact, if you cannot crop into the image then I'd say don't bother cropping at all, because the memory saved is so small.

Warning on Cropping

It is only fair to warn you that, although in my opinion cropping improves the image quality of the DivX rip by a lot, before you decide that cropping is the best way to go, you must consider these four facts:

1. A cropped movie is sometimes harder to convert into another format. This is

because you may need to re-add the black bars to the top and bottom of the movie first or the movie may be stretched out of shape.

2. A cropped DivX movie will play at the wrong aspect ratio in PowerDVD and some other Video players. On the other hand, Media Player, MicroDVD and many other players will play back a cropped movie perfectly.

3. VCD's and SVCD's cannot be cropped if they are to be played in a standalone DVD player because it will not accept them.

4. Finally, Mpeg-4 files (and DVD files for that matter) have problems playing back on some hardware if they are not encoded in sizes that can be divided by 32. This means the Matrox G400 or the Nvidia GeForce would probably have problems outputting it to TV. This TV out problem is associated with the Mpeg-4 codec and does not apply to most other codecs.

As an example:

A 528 pixels wide size divided by 32 = 16.5. This is not a multiple of 32 and so may have trouble.

But a 576 pixel wide size divided by 32 = 18. This is a multiple of 32 and will play back perfectly.

DON'T BLAME IT ON THE BITRATE!

If you find you start to encode a movie and it looks like the picture below it is nothing to do with the compression. It wouldn't matter if you used 6000kbps or 10kbps it wouldn't get rid of these lines. The problem is that Flask, or whatever decoder you are using is unable to decode your DVD correctly!



Since TV screens are built out of lines, the DVD has a code inside it that tells it how to output those lines to the screen and in what order. If the order of the "fields" is incorrect you will get the annoying combing effect we see above. So far there is not complete solution to this. You can try resizing the picture smaller or you can use Flask Mpeg's deinterlace filter. There is also a super slow annoying way to fix this using DVD2AVI. For more information this interlace subject check out the information in my appendix called: "Video Formats: NTSC & PAL / Telecine".

DeCSS ARTIFACTS

As you know, DVD's have a content scrambling system (CSS) which makes it difficult to copy. These require code keys to decrypt. If you use the wrong key you will end up with a corrupted file. Most ripping software automatically finds and uses these keys now, but it is always a good idea to check your decoded DVD by playing it on your computer DVD player, just to make sure it is correct before you convert it to something else. You will know if it's corrupt because you will get loads garbage or green / pink blocks like in the picture below:



As you can see, it is not only 'dead people' this kid sees =o).