
**VicBond007's Guide to
Working with DVD Footage
for Use in
Anime Music Videos**



Copying a DVD

The first step to allow you to use DVD footage as your source, is to copy the DVD to your hard drive. Commercial DVDs are protected by a Content Scrambling System, abbreviated as CSS.

CSS is a set of 5, 2-digit hexadecimal keys that are only supposedly readable by set-top DVD players, and DVD player software. The purpose of this scrambling code is to prevent you from directly accessing the contents of the disc on your pc, otherwise you could easily make a perfect digital copy by just dragging and dropping the contents of the disk to your computer, or decoding the video stream directly off the disc to an AVI or some other file format.

If you tried to decode the contents of a DVD directly off the disc, the picture would be unrecognizable and scattered with various pink and green boxes. If you try to drag the .vob files off the disc, in most situations, it will transfer, however the data will just be unreadable and appear corrupt. If you want to know all the geeky information about CSS, then you can find out all about it at the following address:
<http://www.alleged.com/mirrors/DVD-CSS/>

The solution is to copy the DVD to our drive, and remove the encryption in the process. There are several freeware programs out there to let us do this. We'll be using DVD Decrypter by Lightning UK. This will let us copy our DVD, as well as remove CSS and Macrovision, a visual anomaly deliberately put into the disc to screw up VCRs, making it almost impossible to record a DVD to a tape (it will also cause problems with many capturing cards, as well as cause difficulty if you try to output your video back to tape.)

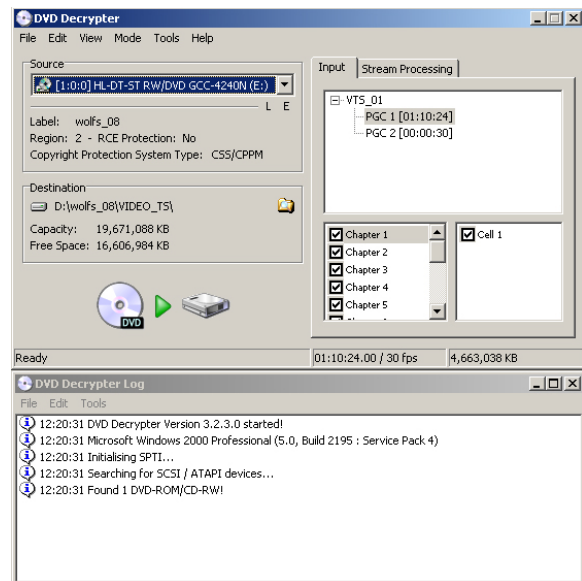


Figure 1: DVD decrypter version 3.2.3.0 detecting the presence of my laptop's DVD drive, as well as the fact that Wolf's Rain: Disc 8 is in the drive.

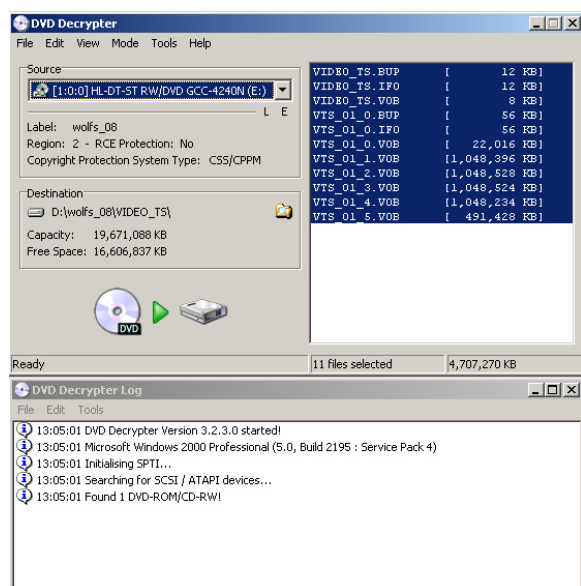


Figure 2: DVD Decrypter as it starts up for the first time.

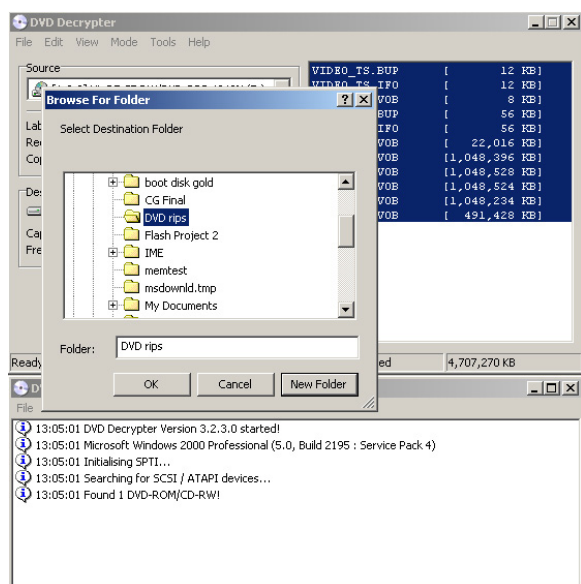


Figure 3: Choosing a destination folder. If you didn't make one already, or don't want to use the default folder, use the "new folder" button. In this example, I will be extracting to "DVD rips".

Choosing Your Region

Depending on where your DVD is from, you may need to modify the firmware on your DVD-ROM before you can read the disc. Even though we are not playing the DVD, a PC DVD-ROM drive is required. A DVD can not be ripped with a basic CD reader/writer drive.

When you buy a DVD that's content could potentially be distributed in other countries, the DVD is flagged with a region number. If your drive does not match the number, it will not play. If you live in North America and have purchased a legitimate region 1 (North America and Canada) release, then this step is not needed. If you've imported your DVD from Japan, the disc is most likely coded region 2, and will require modification to your drive.

Flashing procedures are different for each drive, so be sure to visit <http://forum.rpc1.org/portal.php> where you can download "modified" firmware for your drive (assuming that it's available) as well as get instructions as to how to flash your drive. Follow them closely. A bad flash can destroy your drive.

Setting Up DVD Decrypter

Download and install DVD Decrypter from its website at www.dvddecrypter.com. Put your DVD in your drive and fire up DVD decrypter. You should get a screen similar to **Figure 2**. Don't worry if the contents to the right look very different, it will vary per disc. For now, click on the yellow folder icon in the destination box and specify where you want your rip to go. DVD Decrypter assigns a new folder for each disc, but you may want to change that directory and/or drive to something else.

Recommended Ripping Settings for DVD Decrypter

In almost every case, the main program on the DVD (the episodes themselves) will be lumped into the same PGC.

- Content on DVD is generally arranged as cells (video/audio objects) inside of a Program Chain (PGC: group of cells that play in a certain order) inside of a Video Title Set (VTS: sets of unassociated PGCs). This means every thing to the dvd author, and nothing to us, just understand that what we want is usually in one PGC.

To minimize the number of extras we're copying over, we only want to select our main PGC. DVD Decrypter usually does a good job of automatically selecting this for us, as it's usually the biggest PGC on the disc.

Usually the default settings on programs are pretty bad, but fortunately DVD Decrypter has made some good choices for us. There's only one setting we want to change, so do so by going to the tools menu at the top, click on settings, and click on the "IFO Mode" tab. Near the middle, there is an option for file splitting with a drop down menu to the right of it. It should read "1 GB". Change this to "AUTO". DVD2AVI can get confused when it's fed info for .vob files that are chopped up more than they should be. This setting MAY produce a file larger than 4GB, so be sure that your destination drive is NTFS and not FAT32.

Click OK, go back to the main window, and click the icon near the middle on the left depicting a DVD, a green arrow, and your hard drive. This will open a new window and begin the decryption/copying process. You will be prompted when the process is done. Close DVD Decrypter. We're done ripping.

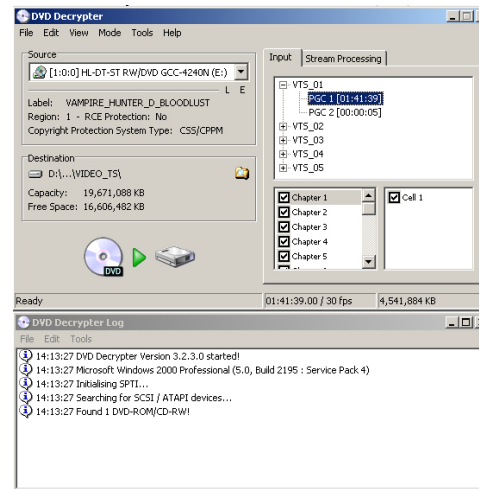


Figure 4: Vampire Hunter D: Bloodlust has a lot of VTS tracks, fortunately DVD Decrypter already found what we're looking for.

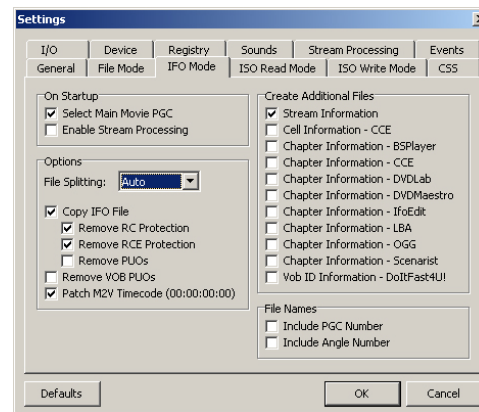


Figure 5: File Splitting on auto. The other settings have their place, but that place isn't here.

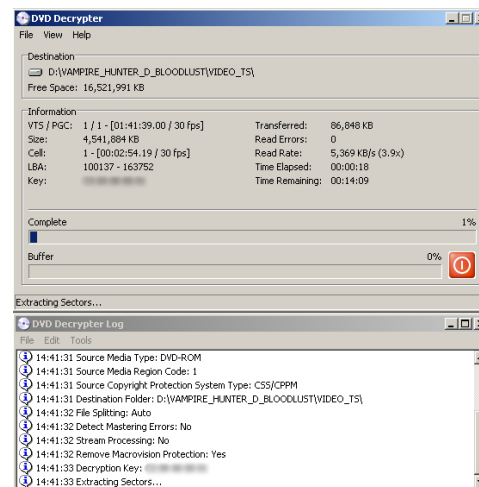


Figure 6: Gone Ripping.

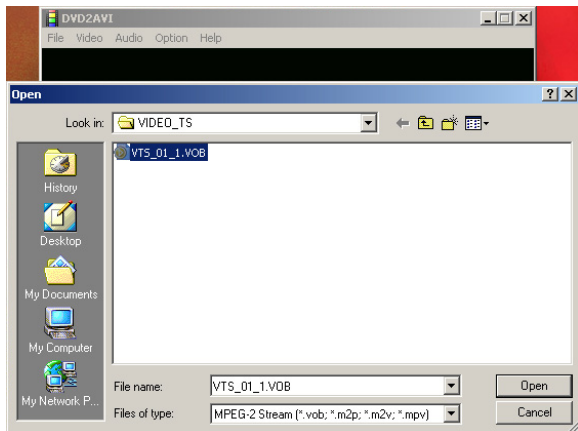


Figure 7: Selecting the unencrypted .vob that DVD Decrypter created for us.



Figure 8: Interlacing. More trouble than it's worth. If you think it looks bad now, wait until you're done editing it.



Figure 9: Marking off an area to encode. Note the shading on the timeline.

Prepping the .D2V File

The next step is to use DVD2AVI to generate a .d2v file. This file is only a few kilobytes in size and tells avisynth (we'll get to it, don't worry) what stream to be looking at inside each .vob.

- Data on a DVD is stored in video object files called .vobs. DVDs can have multiple angles, subtitle tracks, and audio tracks, so we need DVD2AVI to tell us what's important in the .vob files we ripped.

Download DVD2AVI from:

<http://arbor.ee.ntu.edu.tw/~jackei/dvd2avi/>
 unzip it to a directory (Mine's in C:/DVD2AVI), and make a shortcut if you want (DVD2AVI does not use an installer). Run the program, and then go to the file menu and select "open". Browse to the folder you ripped to, and inside it is a VIDEO_TS folder. Go in that, and if you split your files using the "auto" setting, there should only be one big VOB file in there. Open that, and hit OK on the dialogue that pops up afterwards. Now you can seek through your ripped footage, and if you wanted to, mark off areas and pull your clips right from here. The problem with doing this is that you can only seek by I-frames (frames that are re-drawn entirely that usually occur only once or twice a second) and your footage will most likely be interlaced. Unless you REALLY know what you're doing, and want to put up with the headache, I suggest taking a few more steps to postprocess your footage, deinterlace your clips, and make them easier to work with. If your .vob has ads at the end or other extras, you can crop that area out by dragging the playback head to the point where the undesirable content starts, and hitting the button to the far right shaped like a bracket. The area before the playback head should be blue on the timeline, and white in the area after the playback head.

Saving the .d2v

Now that you've marked off the area we'll be reading from, go to the file menu and select "save project" (not "save .avi"). Specify a location and save the file as something.d2v. It's very important that you remember what you named the file, so keep it simple.

A new window will show up to the right of the main DVD2AVI window with various stats, none of which we care about except for the time remaining which is shown at the bottom. When it says "FINISH" then the process is done, and you can close out of DVD2AVI. You will not be able to see the process in action so this is how you will be notified when it is done. It will take about 5 minutes to build a project file for your average 85 minute DVD.

Preparing Your Computer for AVISynth

We're going to need to install a few more things before we can go any further. The first is Virtualdub by Avery Lee, and is available here: <http://www.virtualdub.org/>

Virtualdub is a totally free utility for quickly recompressing video clips, applying filters, and performing remedial cut and merge operations. Download the latest version, unzip it, and make a shortcut in your start menu if you so desire. You're going to be using this program a lot.

Now we are going to download AVISynth from <http://www.avisynth.org/> AVISynth is a script-based program that uses 3rd party filters to pass video data off to applications that normally would not be able to read the video. You can also apply various effects and filtering techniques to clean up your footage and make it more usable. Install AVISynth and be sure to associate .avs files with notepad when prompted by the installation.

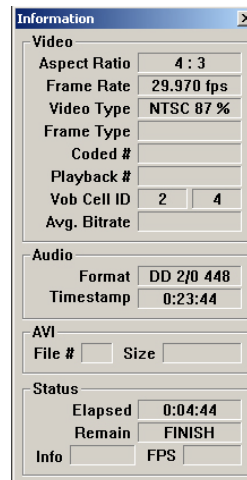


Figure 10: Our project file is done, but the fun is just beginning.



Figure 11: Virtualdub 1.5.10. Soon to be your new best friend.

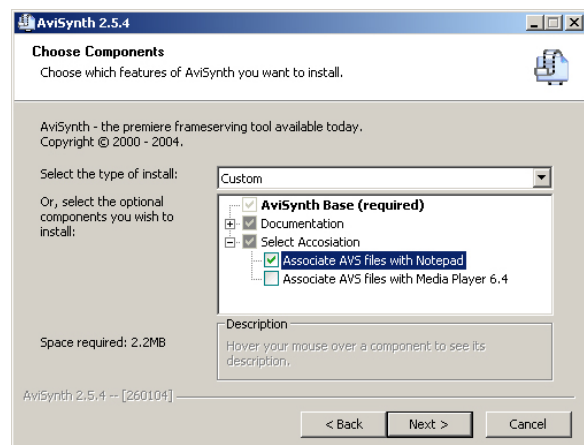


Figure 12: Installing AviSynth 2.5.4

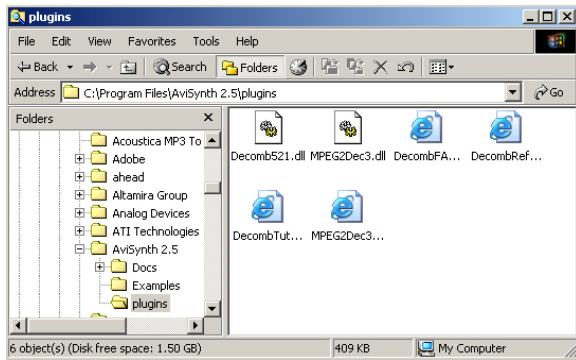


Figure 13: Our AviSynth plugin directory after unzipping our two plugins.

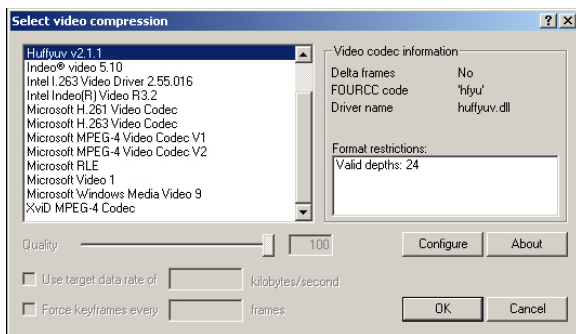


Figure 14: Some, though definitely not ALL the video codecs that you'll probably run across when working with digital video.

Installing Plugins for AVISynth

There are hundreds of plugins for AVISynth to use, and good thing, because the basic features of AVISynth are very limited. For our purposes, we will use only two plugins, Mpeg2Dec3, and Decomb521.

Download Mpeg2Dec3 by Nic from:

<http://nic.dnsalias.com/>

and unzip it into the plugins directory created during your installation of avisynth (C:\Program Files\AviSynth 2.5\Plugins). Take note that the parameters list for the plugin will also be installed to that directory, so if you would like to make any changes to the script we are about to write, you will need to refer to the documentation here.

Next, you're going to download Decomb5.2.1 by Donald Graft from <http://neuron2.net/> and unzip it to the same directory as you did for mpeg2dec3. Before we get scripting, there's one more thing we have to take care of. Mpeg2dec3 processes the .d2v we're going to send it, in YV12 colorspace. What that means isn't important to us, just know that we need to install some codec that will provide us with a YV12 Video for Windows (VFW) filter.

- A codec determines how a piece of media is compressed and decompressed. An .avi file is not a format, as much as it is a container for different types of valid media. Video can be compressed with all sorts of codecs like huffyuv, divx, xvid, mpeg4, or indeo. Likewise, audio can be compressed to formats such as mp3 or ulaw. These formats may even have different methods of encoding and decoding, but they all end up inside the .avi package. Keep your installed codecs to a minimum to keep your system running well.

Installing Video Codecs

For right now, there are going to be three video codecs that we have to install to be able to prepare our footage. As mentioned before, the Mpeg2dec3 plugin uses YV12 colorspace, so we're going to need to install Nic's build of Xvid from here:

<http://nic.dnsalias.com/xvid.html>

We will not actually use this in the process of acquiring footage, but it needs to be installed in order for Virtualdub to read the .vob file through the script we will soon write.

The next codec we'll need is DivX 4.12. This is an older version of DivX, and in my opinion, vastly superior to the bloated, incompatible mess than DivX 5.xx has become. Finding it can be tricky, but for now you can get a copy from: <http://www.schattendal.com/help/divx4.html> DivX4 is at the bottom. Download and install this codec, answering "yes" when it asks if you want to use DivX4.12 to play back DivX 3 files. This process will remove DivX5 from your system. If you need it to view a DivX 5 file, reinstall DivX5. DivX 4 and 5 can not co-exist without some fancy tricks that aren't worth going into.

The last codec we'll need is huffyuv, a lossless codec that can be recompressed infinitely without any perceptible loss in quality. It works as well as uncompressed video, while taking up less space, and thus makes playback easier as well. Download it from:

<http://neuron2.net/www.math.berkeley.edu/benrg/huffyuv.html>

Unzip it to a folder, then navigate to that folder, right click on huffyuv.inf, and select install. A "Copying files" dialogue box should pop up, though the process only takes a second and is completely automated. The codecs we need are now installed.

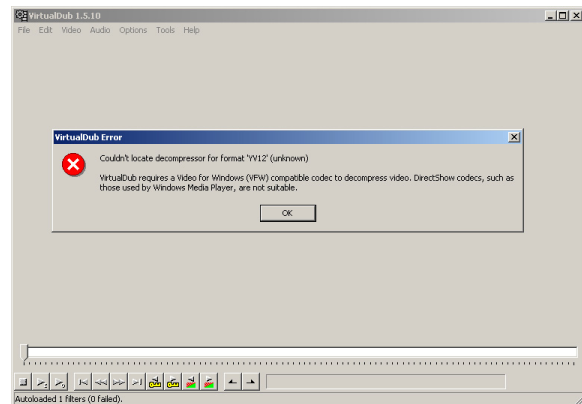


Figure 15: Error message given by Virtualdub if you try to open an AviSynth script using Mpeg2dec, without installing a Video for Windows YV12 filter. DivX 5.xx installs a YV12 filter but you should have overwritten DivX 5 with DivX 4.12 before getting to this point anyways.

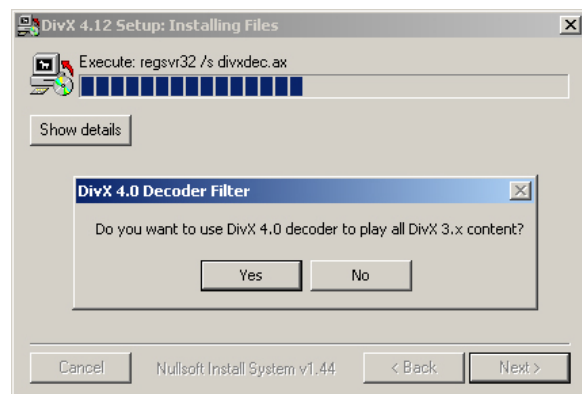


Figure 16: Unlike DivX 5.xx, DivX 4 asks you if you want it to take over. Say yes. DivX 3 is weak.

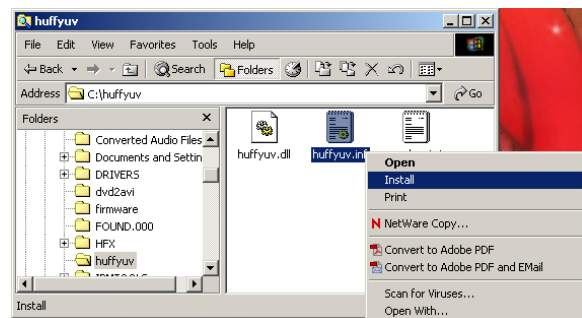


Figure 17: Installing huffyuv.

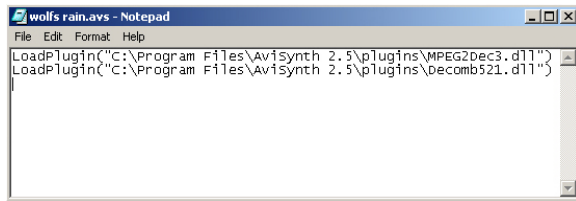


Figure 18: Loading 2 plugins with AviSynth. If your install path/file names are different, adjust accordingly.

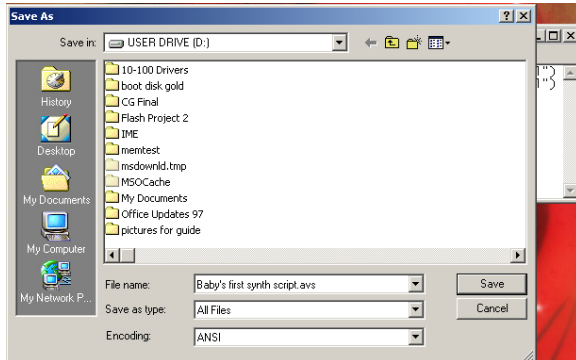


Figure 19: Saving our script. Note that the file type is “all files” and not “text file”. Yes, it matters.

Scripting with AviSynth

Loading Plugins

AviSynth is script-based. It does not have a graphical user’s interface, so all of our work will be done in notepad. We only need to go over a few commands for our purposes, however, every plugin has a multitude of possible settings and configurations. If you plan on deviating from this guide at all, reading the help documents that came with the plugins is a definite must.

Open up notepad. The first thing we have to do is load the plugins we want AviSynth to use.

The syntax for this is:

LoadPlugin(“absolute location”). For our case, we would enter the following code:

```
LoadPlugin("C:\Program Files\AviSynth
2.5\plugins\MPEG2Dec3.dll")
```

```
LoadPlugin("C:\Program Files\AviSynth
2.5\plugins\Decomb521.dll")
```

See figure 18 to see how this should exactly appear in your script. This tells AviSynth to load both plugins that we downloaded. Obviously, if you’ve installed to another location, you would enter that location instead. As you download updated versions of the plugins, the names of the .dll files may change, so pay attention when writing your scripts. Also, pay attention to any stray spaces, unintentional carriage returns, or missing symbols such as quotation marks and parentheses. You are programming on a very basic level, and there is no room for mistakes.

Go ahead and save your work by choosing “save as” from the file menu, and selecting the “save as type” of “all files”. Name your file whateveryouwant.avs.

Loading Footage with AviSynth

The next step is to tell avisynth what footage to look at, and subsequently pass to virtualdub.

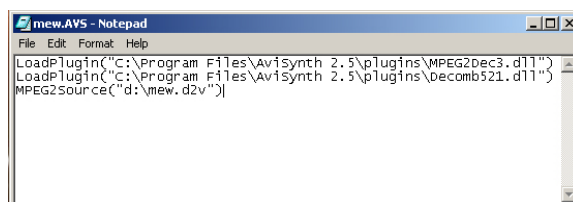
The Mpeg2dec3 plugin does just that. The base syntax for Mpeg2dec3 is:

```
MPEG2Source("full path/whatever.d2v")
```

See figure 20 for an example of this line of code being added to our existing script.

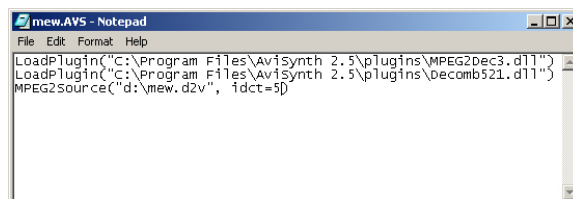
Mpeg2dec3 supports other variables, and we will use one or two. One variable that's important to us is the variable "idct". It stands for Inverse Discrete Cosine Transformation and affects how accurately the mpeg2 file inside the .vob is read. More accurate methods may be slower, buggier, or both. Higher settings do not equal higher quality. We only need to adjust this parameter if the computer you're on supports SSE2 processor instructions, such as a Pentium 4. If so, place a comma outside of the end quote for the file path, and add the line `idct=5` before the end parentheses. See figure 21 for an example of the final line of code. Omitting this variable will not hurt the process any.

Mpeg2dec3 also supports deblocking and deringing through the "CPU" variable. Deblocking will remove some artifacts from poorly compressed DVDs, whereas deringing tries to remove rainbow artifacts caused by poor analog source (your digital DVD had to be mastered from something!). You can specify horizontal or vertical, as well as chromatic or luminance deblocking and deringing. If you have to use it, you might as well go all the way. For maximum deblocking precision, use the value of 4. For deblocking and deringing, use 6. Deringing will slow down your processing speed a LOT, so use it only if you notice rainbows in areas of high contrast. If your source is interlaced, you'll also have to set the variable `iPP` to "true" to enable field based postprocessing.



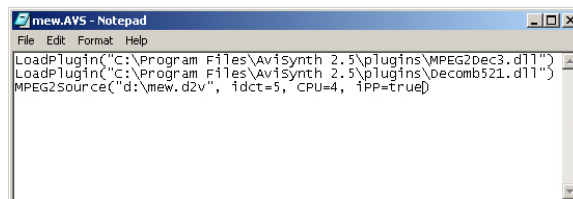
```
LoadPlugin("C:\Program Files\AviSynth 2.5\plugins\MPEG2Dec3.d11")
LoadPlugin("C:\Program Files\AviSynth 2.5\plugins\Decomb521.d11")
MPEG2Source("d:\mew.d2v")
```

Figure 20: The most basic (but most essential) parameter for MPEG2Source.



```
LoadPlugin("C:\Program Files\AviSynth 2.5\plugins\MPEG2Dec3.d11")
LoadPlugin("C:\Program Files\AviSynth 2.5\plugins\Decomb521.d11")
MPEG2Source("d:\mew.d2v", idct=5)
```

Figure 21: Optimizing idct for the pentium 4. All variables for plugins in avisynth scripts are separated with a comma, followed by a space, and occur after a string (words in quotes) when applicable. If your CPU doesn't support SSE2, or you're not sure, it'd be best to not even invoke this variable at all.



```
LoadPlugin("C:\Program Files\AviSynth 2.5\plugins\MPEG2Dec3.d11")
LoadPlugin("C:\Program Files\AviSynth 2.5\plugins\Decomb521.d11")
MPEG2Source("d:\mew.d2v", idct=5, CPU=4, iPP=true)
```

Figure 22: Adding field based deblocking capabilities to the above script. If your source DVD is progressive (Many region 2 widescreen DVDs tend to be) then you can omit the `iPP` variable, as the default setting is "false".



Figure 23: Pixellated scenes such as this one from the opening of Tokyo Mew Mew on the first disc could use some deblocking, however mpeg2dec3's method makes little change to the original image. We're better off filtering through virtualdub later.



Figure 24: Examples of video encoded through different telecine procedures. On the left, Blue from Wolf's Rain shows off her legs, as well as properly telecined video through 3:2 pulldown. You can't see it well in the scaled-down image, but the last frame is slightly interlaced as well. On the right, Mew Ichigo shows us not only poor telecine (1 frame in 5 is interlaced) but if you look closely, the background is properly telecined, just not in sync with the rest of the animation! Background and character animators usually work separately, and these are the things that can result from a group effort gone bad. Fortunately, the background isn't moving much, so if we can just properly deinterlace Ichigo, our footage should be OK.

Telecined Interlaced Video

Aside from a progressive scan DVD right out of the box, the next best scenario is that the content of your DVD was originally 24 frames per second (23.97 to be exact) and then converted to NTSC spec (29.97 frames per second) through a process called telecine.

Film, and a surprising amount of animation, run at 24 frames per second. The television system that we use in America, Canada, and a lot of other places in the world (NTSC) runs at 29.97 frames per second. The telecine process is where after 4 frames in a 24fps sequence, another frame is added where the even lines are the content of the previous frame, and the odd lines are the contents of the next frame. This makes 1 out of every 5 frames interlaced. There is also a method called 3:2 pulldown which replaces the 4th frame with an interlaced frame as well, though it is less common in animation (even if it is the "right" way to telecine video). Confused? Go visit aluminum studios at: <http://www.aluminumstudios.com/digitalvideo/advanced/interlaced/interlaced.html> He has a lot more space to explain this than I do. The important thing is to know that you should be looking to see if only 1 out of every 5 frames are interlaced or not. It should be a perfect sequence (p,p,p,p,i,p,p,p,p,i...etc.) If not, check for evidence of 3:2 pulldown (3 progressive frames followed by 2 interlaced frames). Either way, the video is telecined.

Even though a lot of animes (with the exception of movies) are not drawn at 24 fps, they are very often drawn at 12fps and each frame is doubled up to the rate of 24fps. The scene is then telecined to get up to 30(29.97)fps.

Pure Interlaced Video

When working with officially licensed DVDs, this type of interlacing is usually not present. Pure interlaced video occurs when every frame that contains motion is interlaced. This is evident in DVDs that were mastered by capturing footage off VHS tapes or other analog sources, and is very common in bootleg (HK) DVDs. This has to do with how NTSC signals are transmitted and what creative liberties you can take when preparing footage to be displayed on a television. Even if your broadcast is progressive scan, when captured to an analog source, it will be interlaced during any frames that have motion. Many times, DVD authoring studios have access to the original cells, and can construct properly telecined video for their DVD.

Deinterlacing this type of footage is not going to produce as good a result as deinterlacing telecined footage. We'll go over 2 methods a little later on. One will blur the two interlaced fields together creating a ghosting effect, the other involves cutting the resolution of the video in half. Neither looks very good, so it's highly advisable to stay away from bootleg/poorly mastered DVDs.

Checking Your Footage

Open up Virtualdub and using file/open, load up the .avs script we've been writing. Our video should load fine. Scroll to a scene with lots of action, and use the right arrow on your keyboard to advance frame by frame, and check for evidence of telecined footage (a sequence of progressive and interlaced frames) or pure interlaced footage (every, or almost every frame is interlaced).

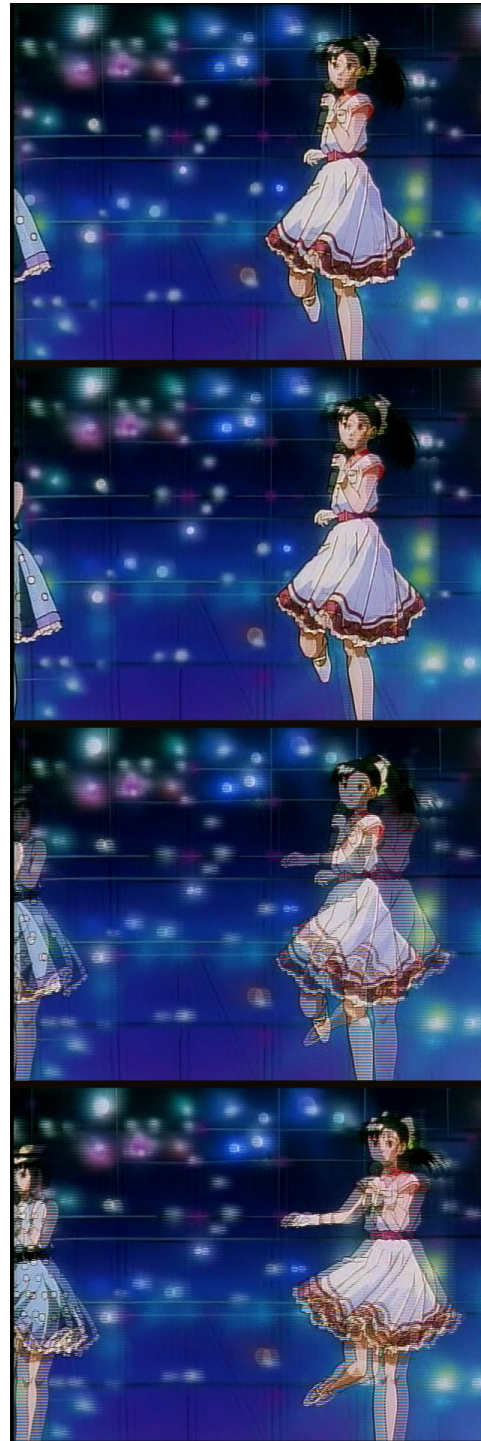


Figure 25: This crap-tacular authoring job on the bootleg Idol Defense Force Hummingbird DVD (Not that the author owns it...) is a perfect example of pure interlaced footage. Avoid DVDs like this at all costs.


```

wolfsrain.avs - Notepad
File Edit Format Help
LoadPlugin('C:\Program Files\AviSynth 2.5\plugins\MPEG2Dec3.dll')
LoadPlugin('C:\Program Files\AviSynth 2.5\plugins\pcomb521.dll')
MPEG2Source('d:\wolf08.d2v', 'idct=5')
Telecide(order=1)

```

Figure 26: Applying the telecide operation. Our Wolf’s Rain DVD is region 2 from Japan, so the field order has to be specified as 1 (top fields first) by specifying `order=1`.



Figure 27: Loading our footage with the wrong field order. In this example, I loaded our `order=1` footage as `order=0` instead. The result is a blocky, striated mess.

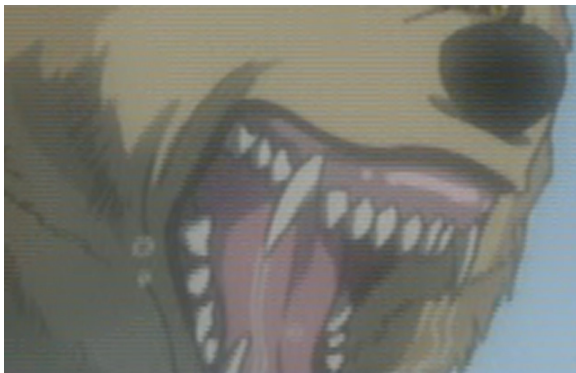


Figure 28: The same frame as in figure 27, only with the correct field order. The edges are much smoother and the cleanliness will be noticeable in the final project.

Using Telecide to Isolate Progressive Frames in a Telecined Sequence: Field Order

Telecide is an operation that analyzes our video sequence, looking for interlaced frames. It then uses the progressive frames before the suspect frame, and after it, to accurately reconstruct a progressive image. Telecide does not alter the frame rate.

If our original footage was telecined up to 30fps, we will use the Telecide operation to remove the interlacing artifacts and reconstruct the progressive images. The basic syntax to load this operation is `Telecide()`. All the variables (and there are a lot!) will go inside the parentheses. While omitting variables can/will invoke their default values, there is one variable we have to set before we can save our script.

Telecide needs to know the field order of our footage. Some footage is read even lines first, odd lines second, and some is read the other way around. The variable is named “order” and giving it a value of 0, will load the even fields (bottom) first, while 1 will load the odd fields (top) first. If your DVD is from North America, use `order=0`. If it is from Japan, use `order=1`. If you’re not sure, pick one, save your script, and load it in virtualdub. If the edges of things in action scenes look jagged, as if you were scaling up low resolution video, then your field order is backwards. Likewise, if the image looks perfectly smooth, then you guessed correctly.

Save your script, and open it in virtualdub if you want. We’re far from done, but this is a good way to check your scripting progress.

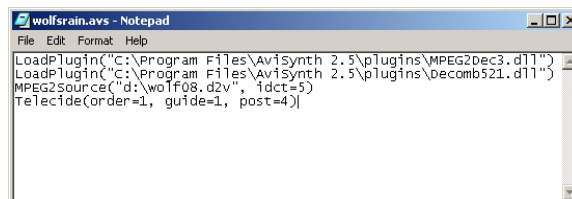
Using Telecide: Setting the Guide

Another variable that will help telecide isolate frames that need to be fixed, is the variable 'guide'. By default, telecide uses no guide, and looks at every frame. If we know that our footage was telecined, we can set guide=1 so it can count frames and determine which ones it should spend more time analyzing. Guide can also be set to 2 or 3, but these modes are unimportant to us at the moment as they are both concerned with video encoded in the PAL format.

Postprocessing with Telecide

By default, when telecide runs across an interlaced frame, it checks around it, looking for the closest progressive frame, and then replaces the interlaced frame with that. If you have footage that is telecined, though inconsistently (almost every DVD I've run across is this way) then you may want to change this method.

The variable for this is 'post' and its default value is 2. Changing it to post=4 will result in telecide using surrounding progressive frames as a guideline to deinterlace the current interlaced frame. This is recommended for anime DVDs that frequently cut in the middle of frame sequences. In a later step, we will be removing these frames anyways, so if you're feeling really lucky, you can set post=0 for telecide to simply "tag" the frame to be removed. However, there's always a chance of frames slipping through, so it's better to be safe than sorry. Even on a grand scale (encoding the whole disc) the time difference between 0 and 4 is minimal. There is usually no visual quality tradeoff when employing post=4 over post=2.



```
wolfsrain.avs - Notepad
File Edit Format Help
LoadPlugin('C:\Program Files\AviSynth 2.5\plugins\MPEG2Dec3.dll')
LoadPlugin('C:\Program Files\AviSynth 2.5\plugins\Decomb521.dll')
MPEG2Source('d:\wolfr08.d2v', idct=5)
Telecide(order=1, guide=1, post=4)
```

Figure 29: Adding guide=1 to help telecide figure out which frames to focus on. Post=4 has also been added to create a more reliable, intermediate picture during interlaced frames.



Figure 30: Different uses of 'post' with telecide. The top frame is post=0, the middle frame is post=2, and the bottom frame is post=4. Note that with such good source footage, there is no difference between 2 and 4.

```

mew.AVS - Notepad
File Edit Format Help
LoadPlugin('C:\Program Files\AviSynth 2.5\plugins\MPEG2Dec3.dll')
LoadPlugin('C:\Program Files\AviSynth 2.5\plugins\Decomb521.dll')
MPEG2Source('d:\mew.d2v', 'fdct=5')
Telecide(post=4, order=1, guide=1)
Decimate(quality=3)

```

Figure 31: adding decimate() to our script and increasing the precision of the scanning process by modifying the quality variable.



Figure 32: 12 frames taken from a sequence that has been run through telecide and decimate. This sequence lasts half a second, and there are no interlacing artifacts to be seen.

Decimate: Eliminating Telecined Frames through AviSynth

Now that we've isolated the telecined frames, it's time to save some space and increase the quality of our final output (you'll see why later) by removing the telecined frames using the Decimate() operation.

Decimate() is added on a new line underneath Telecide() in our script. Decimate supports quite a few variables as well, but there is only one variable that we ever really need to touch, since the default values, for the most part, are optimized for our purposes.

By default, Decimate searches ahead by 5 frames, then removes the frame that is most like its surrounding frames. This is why it was so important for us to use telecide in our script, as it creates a situation with 3 (as opposed to 2) or 3 (as opposed to 4) identical frames in a row. With pure film sources without identical frames, this creates one. Regardless of the source, there's an extra frame in the sequence and Decimate() will remove it. This process drops the frame rate down to 24fps (actually 23.976) and when coupled with telecide(), ensures that virtually all of the frames are progressive. This process is called Inverse Telecine.

Technically we could be finished at this point, but first it is recommended to use the variable "quality" and set it equal to 3 (quality=3) inside the parentheses for Decimate(). This ensures the highest quality of accuracy while searching the video stream, as it includes chromatic differences in the decision making process as well. The default value (2) does not do this. Changing this setting makes a very big difference with animation.

Problems with the Decomb Plugin

Telecide and Decimate work by actually looking at the contents of the video stream on a frame by frame basis. For it to work properly, the 30fps sequence must have at least 24 progressive frames in it. For pure animation, this is almost always the case.

We run into trouble with modern animation that combines CG (Computer Generated) elements, with traditional cells. Nowadays, it's not uncommon for animated backgrounds, vehicles, and special effects to be done on computer instead. Because the computer is doing the work, these elements are usually done at a full 30fps.

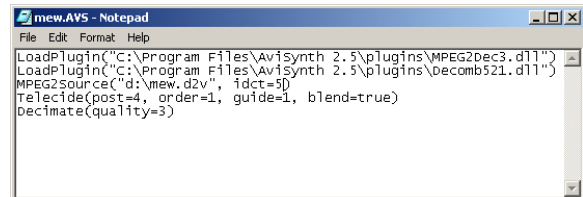
These 30fps elements look nice, but confuse decomb a lot during sequences that combine CG and cells. This is very evident in magical girl transformation sequences where the character is hand drawn, but the backgrounds are rendered in a computer. Since the background is always moving and the character is not, it's tricky for the plugin to find out which is the interlaced frame.

There are 3 solutions, none are perfect. The first would be to write a different script for every one of these sequences you want to rip. This creates a lot of wasted time, time better spent editing. Another option is, in the parentheses for Telecide, add `blend=true` to the list of variables. This will blend the interlaced frame with the frame after it, creating a ghosting effect, but smoothing out the image.

By default, Telecide drops the vertical resolution on the interlaced frame by 1/2, then scales it back up, eliminating the interlacing, but adding jagged edges. I recommend this approach, as it's barely noticeable on a tv monitor anyways.



Figure 33: Zoomed image of Mew Ichigo during a transformation sequence. Telecide couldn't figure out which frame to drop, and guessed the wrong one, but it still recognized this frame as being interlaced, so the frame has been scaled down and back up to eliminate the striping.



```
mew.AVS - Notepad
File Edit Format Help
LoadPlugin("C:\Program Files\AviSynth 2.5\plugins\MPEG2Dec3.d11")
LoadPlugin("C:\Program Files\AviSynth 2.5\plugins\Decomb521.d11")
MPEG2Source("d:\mew.d2v", idct=5)
Telecide(post=4, order=1, guide=1, blend=true)
Decimate(quality=3)
```

Figure 34: Adding the blend variable to our script. By default, `blend=false`, and interpolates interlaced frames that slip through as demonstrated above.



Figure 35: Blending two frames together to make the scene easier on the eye. It is harder to work with though, and very noticeable if you slow the clip down. Going with the default setting for blend is probably the best idea since frames like these are few and far between, and only last for 1/24th of a second anyways.

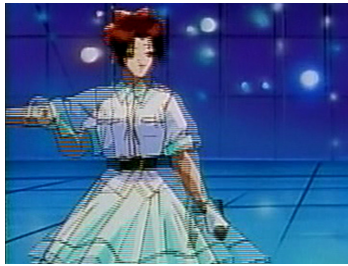


Figure 36: No inverse telecine is going to remove these artifacts. Our only hope is FieldDeinterlace().

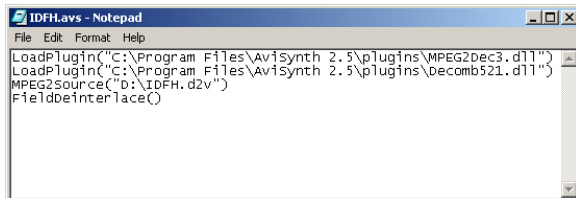


Figure 37: Applying FieldDeinterlace() with the default settings.

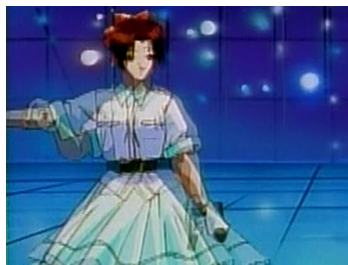


Figure 38: FieldDeinterlace() in action. The scene is ghosted, but at least it's smooth looking.

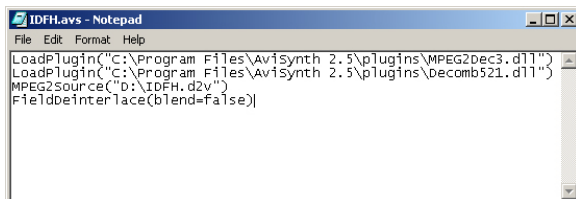


Figure 39: Activating interpolation by setting the blend variable to false.



Figure 40: Blend=false. The edges don't look THAT bad in this scene. your mileage may vary.

Deinterlacing Pure Interlaced Video with the FieldDeinterlace() Operation

If you're unfortunate enough to have to work with pure interlaced material, you can still deinterlace it, but we can not effectively use telecide() or decimate() to clean up the combing artifacts. Instead we will use a third function of Decomb521 called FieldDeinterlace().

The default options for this function are all pretty good for our needs. By default, FieldDeinterlace blends each frame with the frame that occurs after it, regardless of whether or not it detects any interlacing artifacts.

If you would rather interpolate (scale down, then scale up) the video to eliminate the ghosting, you can by adding the variable 'blend=false'. This will create a lot of rough edges though, and I would suggest against it in most situations.

You can also tell it to analyze the stream and only deinterlace the frames it sees as being interlaced, by adding 'full=false'. If you do this, you may also need to change the variable 'threshold' as well if too many interlaced scenes are skipping through. By default, if full=false, threshold=20, where threshold can be assigned any number between 0 (all frames deinterlaced) and 255 (nothing deinterlaced). Deinterlacing all the frames would give your video a slightly softer look during areas of little-to-no motion, which is not necessarily bad.

I recommend against using full=false and blend=false, but it's a matter of personal preference at this point. Try it out and see which one looks better for your material. For our purposes, we're quite done with scripting at this point. So, play around with changing variables and see what works for you.

Working With Footage

Using Our Script to Generate an AVI that is Easy to Seek Through

Now that we've created a script and you've had some time to play around with your .avs in Virtualdub, you might have noticed something when scanning through your stream...it's slow. Every time virtualdub displays a frame, it first has to apply all the plugin features that we're using. There are also no keyframes, so it takes a lot of guess-and-check dragging to find the area we want.

This next step is a step that very few people do. In fact, I believe I'm the only person who does it because it can have a negative effect on quality, but nobody has ever complained about my video quality before, and I did win AWA Masters using this technique. If the Masters judges can't tell, the general audience probably can't either.

What we're going to do is encode the ENTIRE DVD to a divx file. Not the poorly compressed divx fansubs that you might have downloaded (and edited with, FOR SHAME!) but a rather high bitrate encoding with minimal loss in quality.

Depending on the aspect ratio of your source, there may or may not be an extra step here. If your source footage is 16:9 (widescreen) then we're going to need to add borders to letterbox the video. Your video may look a little horizontally stretched when previewing in virtualdub, this is ok. Without getting into ugly details, a tv pixel is taller than a pixel on a computer monitor, so this image will stretch vertically a bit when displayed PROPERLY on a tv. If the footage looks horizontally squished instead of stretched, then we're working with 16:9 anime.

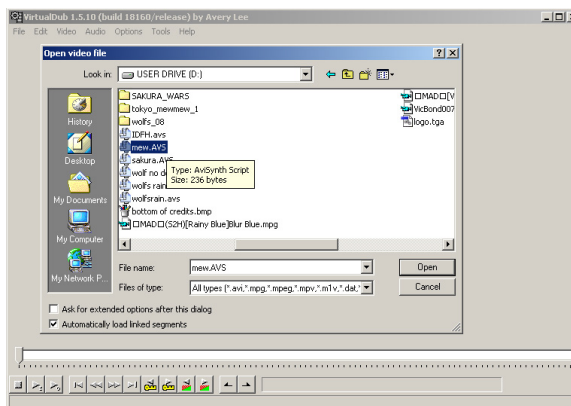


Figure 41: If you haven't been doing so all along, open Virtualdub and using file/open video file, select your .avs script and open it. If you've been following directions, there should be no errors when opening the file.

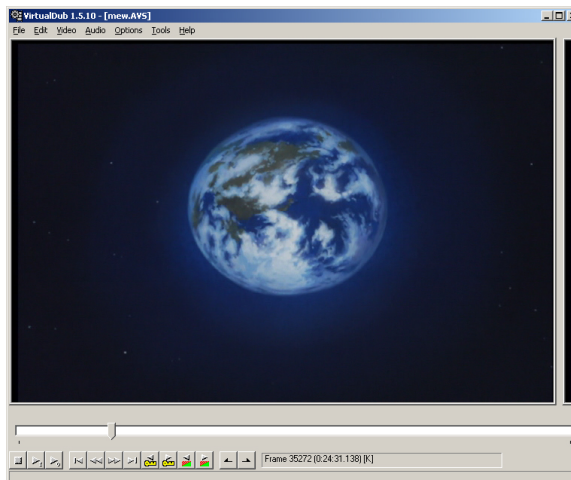


Figure 42: Differences between tv pixels and pc monitor pixels cause the Earth to look a little wider than it should be (our planet IS more circular than this). No cause for alarm though because we know what we're doing...right?

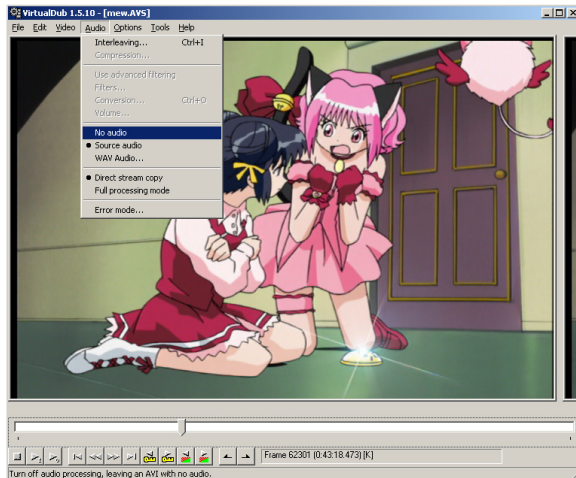


Figure 43: Selecting “no audio”. The video menu is directly to the left of the audio menu.

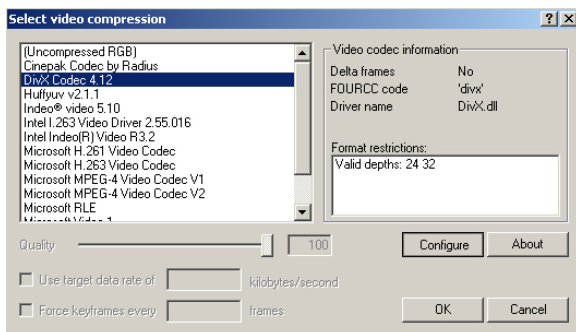


Figure 44: Selecting DivX 4.12 in the video compression popup window.

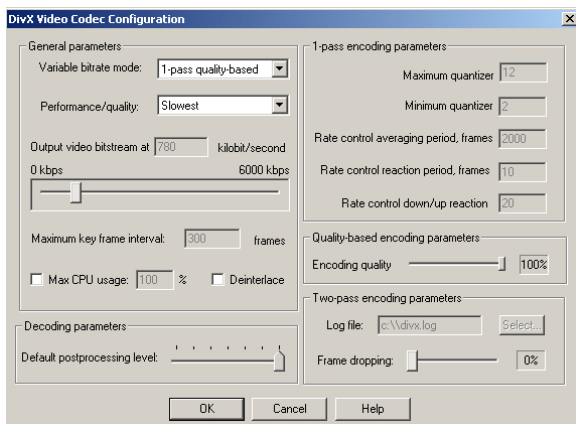


Figure 45: Recommended settings for converting the DVD to DivX. Warning: File size could end up very big (1,200MB/episode) depending on the quality of the original source. Higher quality DVDs will produce smaller divx files because they contain less visual artifacts.

Encoding a 4:3 DVD to DivX with AviSynth and VirtualDub

With our .avs file open in virtualdub, go to the audio menu at the top, and select “no audio”. Even though there is no audio in the stream, virtualdub would have added dummy audio, wasting precious hard drive space and confusing windows media player.

Next, we’re going to go to the video menu and select compression (CTRL+P). A lot of people recommend changing the processing mode from “full processing mode” to “fast recompress”, however I’ve tried this on a number of computers, and there is absolutely no speed gain between the two modes.

Once in the compression dialogue, select DivX Codec 4.12 from the list on the left, and then click the ‘configure’ button on the right. A new window should pop up. What we’re going to do is tell DivX to use all the bitrate it possibly can, but only use as much as it needs. To do this, in the top left, change the “variable bitrate mode” from ‘1-pass’ to ‘1 pass-quality based’, using the drop down menu. Beneath that, in the performance/quality drop down, select “slowest” if it hasn’t already been done for you. To the right, near the middle, there’s a slider for “quality based encoding parameters”. By default it is 85%, meaning DivX will compress the video so that it looks 85% as good as the original, thus saving some file size. We want DivX to try to make the video look as good as possible, so drag this up to 100%. While you’re dragging sliders around, you can also change the ‘default postprocessing level’ slider to full by dragging it all the way to the right. Click OK, and OK again on the original codec window. Go to file/save as .avi, give this disc a name, make sure you have a lot of space, then hit OK. Go to work, or take a nap. This will take a while.

Encoding a 16:9 DVD to DivX with AviSynth and Virtualdub

In the event that you are dealing with widescreen footage, you'll notice that when seeking around in virtualdub, the image looks horizontally squashed. DVD players understand how to adjust this image for a standard tv set, we need to compensate by hand.

There are plugins that allow you to resize footage with avisynth, however they can not be used with the decomb plugin since it needs the ORIGINAL video size to make it's calculations. We will use virtualdub to resize instead.

Perform all the steps necessary for encoding a 4:3 DVD to DivX as mentioned on the previous page, but do not save the avi yet. Instead, go to the video menu again and select 'filters' (CTRL+F). Click the 'add' button and scroll down the list until you see 'resize'. Hit 'OK' and a new window should pop up. What we want to do is add letterboxed borders to our video, while at the same time scaling it properly, AND keeping it so that it will look right when displayed on a tv. To do this, we'll have to first convert the image size to look right on a pc monitor, and then scale it back down to tv proportions. Rather than go into a lot of unnecessary details, just trust me on the numbers here. It may seem like an indirect approach, but it's the best way to be sure you're doing everything you can to make your video look it's best when presented.

For starters, set the new width to 720 (standard television width) and the new height to 405. Widescreen anime is presented in a format called 16:9, meaning for every 16 square pixels wide, there are 9 square pixels high. $(720/16)*9=405$. For the 'filter mode' drop down, select 'bicubic' for the best quality resize.

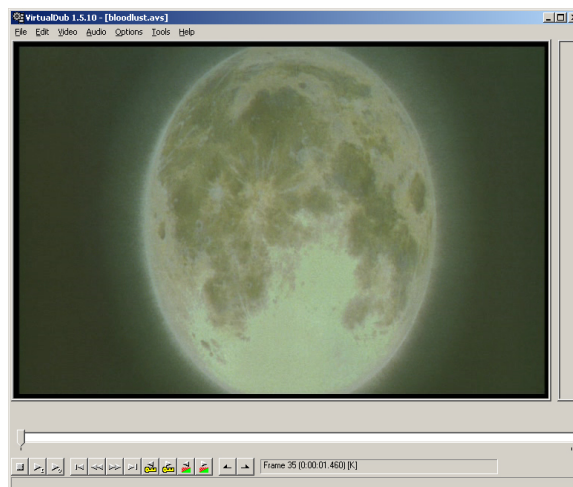


Figure 46: To all of you that used this scene from Vampire Hunter D: Bloodlust, and screwed up the aspect ratio...OUR MOON DOES NOT LOOK LIKE AN EGG!

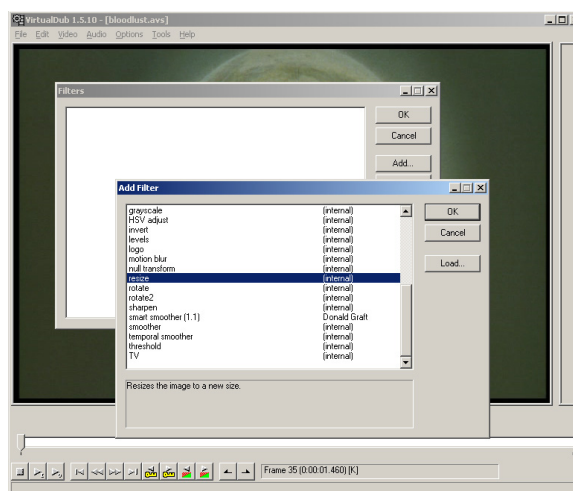


Figure 47: Selecting the resize filter from the list of available filters. It is included with virtualdub, so you don't need to download it like you do with avisynth plugins.

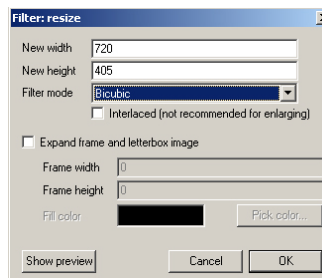


Figure 48: The first step to resizing your 16:9 video. Don't close out of this yet, we're not done.

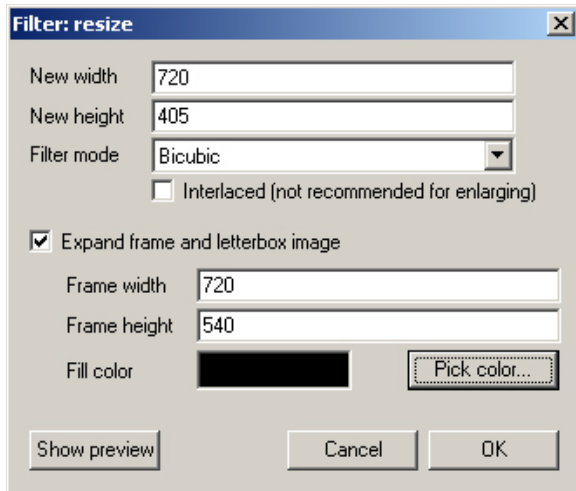


Figure 49: Adding letterboxing. At this point, the video looks perfect when displayed on a computer monitor. We need one more step to prepare it for use on a tv.

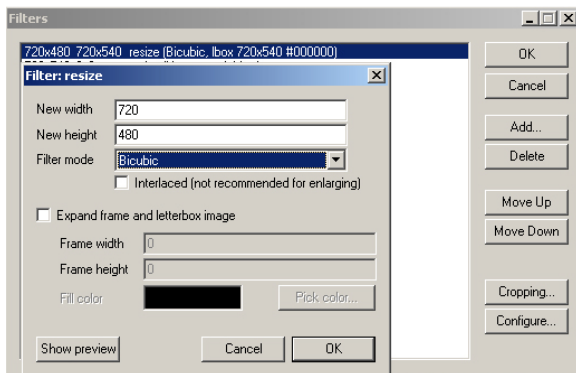


Figure 50: Our second resize. Fortunately, virtualdub lets us use more than one filter on a clip, so we don't have to render this out in two passes.



Figure 51: Now THAT'S a moon!

Next, we're going to need to add our letterboxed borders. However, we've sized the video to proportions that are good only for square pixels, so instead of adding borders, taking the size up to 720x480, we need to add 720x540 borders instead. We do this by clicking the check box next to "expand frame and letterbox image". The fields below it will light up. Leave the frame width as it is and set the frame height to 540.

If you don't trust me, or just want to see how it looks, you can click on the "show preview" button now and scrub around on the timeline in the new window that pops up. The video should look letterboxed, and perfect on your computer monitor. Close the preview window, and hit 'OK'.

We should be back at the filters window now, and our resize filter should be listed along with the settings so that we can see what it's doing at a glance. What we need to do now is resize this video AGAIN, so that it displays itself properly on a television (vertically squished on a pc monitor). Click the 'add' button again, scroll to resize, and click ok. This should be familiar to us by now.

Enter 720 for the width, 480 for the height, and then bicubic for the filter mode. No need to touch anything else, so hit 'OK'. Our resize filter should now be listed twice, each occurrence with different settings. Click 'OK', and we're done setting up our encode. If you haven't configured divx yet, do so now, and then go to file/save as AVI, and go find something to keep yourself occupied. This is going to take a while...

Pulling Clips for Use in Your Favorite Video Editing Program

Now that we've constructed our DivX, and have watched it to make sure it looks good, we can delete our ripped .vobs if we're tight on space. We won't need them anymore unless we ruin our new .avi file. Open your divx4 .avi file in virtualdub and hold down the right arrow key to seek ahead. You'll notice you can scan much faster. Don't hit the left arrow (seek backwards) as DivX is only meant to be read one way. Instead, hold the shift key and use the arrow keys to navigate, snapping to keyframes. You can scan forwards and backwards this way at incredible speed. You can scrub the timeline with your mouse while holding shift as well.

To pull a clip to edit with, seek to the point where you want your clip to start, and click the second button from the right at the bottom of the virtualdub window (if you hold your mouse over it for a few seconds, a tooltip will let you know this is for "mark-in"). Now navigate to the end of your clip, and push the button furthest to the right to mark out. You should now have 2 (though your clip may be too small to see both) little black arrows underneath your timeline.

From here, go to video/compression, select huffyuv, and click OK. There is no need to change any of the settings. If you've added a filter to a previous clip and haven't closed virtualdub in the meantime, go to video/filters and remove the filters since they stay, even when you load different video files. Also, make sure 'no audio' is ticked in the audio menu.

Go to file/save as AVI, name your clip, and watch it go. This is usually a rather fast process and is greatly affected by the speed of your hard drive. These clips are going to be big, but they're going to be the most friendly to edit with, so make sure you have lots of space.

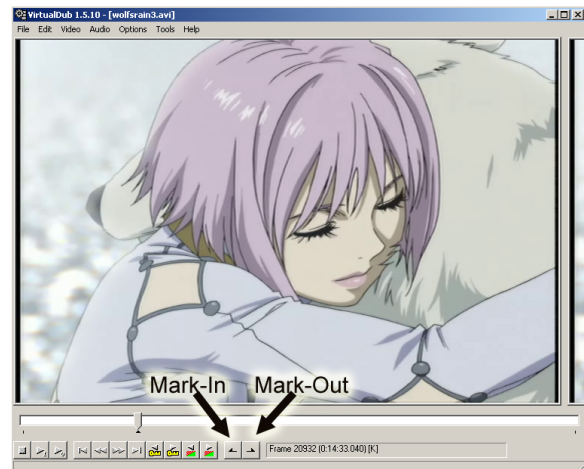


Figure 52: Marking off a clip in Virtualdub. Note the black arrow under the timeline marker.

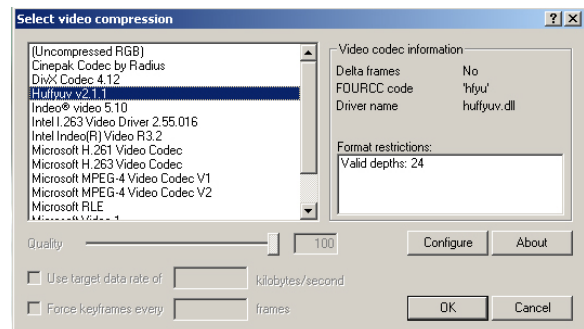


Figure 53: Compressing our clip with huffyuv. doing so means no further quality loss, while maintaining a very high level of editability due to the way huffyuv tracks frames.

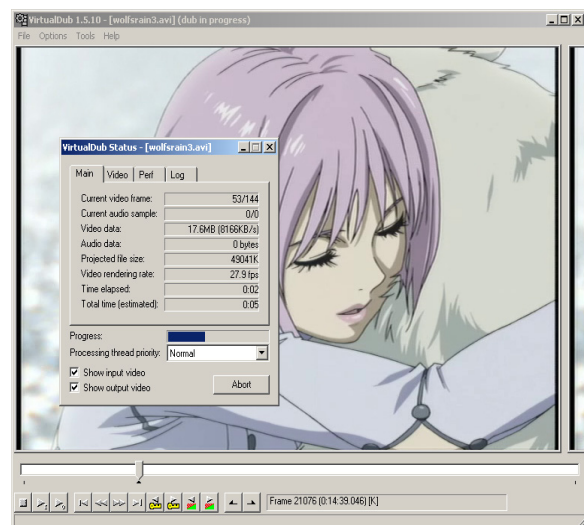


Figure 54: Saving our clip. Once this finishes, we can import this clip into Adobe Premiere and edit with it easily.

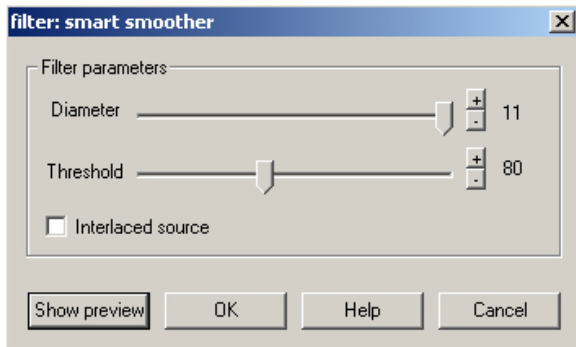


Figure 55: Recommended settings for Smart Smoother 1.1. Your video might need a higher or lower threshold, so be sure to take advantage of the “show preview” button.



Figure 56: This scene needs some serious de-pixelating. No wonder these DVDs were cheap...for Region 2.



Figure 57: Smart Smoother gets rid of a lot of the blocking. Some has still slipped through, but higher thresholds would start to have negative effects on the image.

Cleaning Bad Footage: Heavy Pixellation

Just because it’s a DVD, doesn’t mean the quality is going to be perfect. This is evident in a lot of DVDs released on this side of the Pacific Ocean, and even more so in bootleg Hong Kong DVDs. A surprising number of “professional” studios either have no clue how to make a decent disc, or know that they have your money anyways, so it doesn’t matter to them how high the bitrate is or how precise they encode their streams. There are going to be quite a few times when you need to clean up a clip, and there are two plugins for Virtualdub that we’re going to use. It’s best to use them when pulling clips, rather than when encoding the whole DVD, because using them will make the process go about 20x-30x slower.

We can clean up badly pixellated scenes using Smart Smoother by Donald Graft, available from <http://neuron2.net/smooth.html>. Download the plugin, and unzip it into your plugins directory inside your virtualdub folder. Virtualdub will automatically add it to the filter list the next time it loads.

To use it, load up your footage, mark off your clip, then go to video/filters and click add. Smart Smoother 1.1 should be in your filter list. Select it, and hit OK to bring up the dialogue box. Now would be a good time to hit “show preview” and scroll to a frame that is in your clip. Adjust the diameter setting to the max (11) so that Smart Smoother is set to remove large artifacts. Now use the + and - buttons for Threshold until most of the artifacts go away. Too high a threshold can begin to blur the image and make it look funny, so only use as much as you need. Generally, 75-80 works just fine. Hit OK when you’re done, and save your clip as normal. Be sure to be using Full Processing Mode, otherwise virtualdub will ignore filters.

Cleaning Bad Footage: Rainbow Artifacts

You shouldn't see this problem very much any more, but early DVDs, or current ones mastered from old shows, may be susceptible to an analog artifact known as rainbowing. This occurs either because of poor analog source, poor capturing equipment, or poor cables connecting the two. Rainbows occur on areas of high contrast, such as the black edges of anime characters, and their considerably lighter skintones. A big problem with animation, but one that can be remedied easily thanks to a virtualdub plugin called Smart Smoother IQ by Tim Park.

Download Smart Smoother IQ version 0.6 from <http://www.doki.ca/filters/> and unzip it to your virtualdub plugins directory. Load up your DVD rip in virtualdub, mark off your clip with mark-in/mark-out, and then go to video/filters, then click 'add' to select Smart Smoother IQ.

Maxing out the values won't hurt your image any, aside from dulling the color an imperceptible amount. Most of the time you will need to turn both the diameter and threshold values up all the way to get a visible result. In fact, with some extreme cases, you might want to add the Smart Smoother IQ filter TWICE, with the maximum values both times. No need to check "interlaced source" because you've deinterlaced ahead of time. You should see a very noticeable decrease in rainbow artifacts, at the expense of processing time. Don't be surprised if your clip encodes at 1 frame per second or slower if you've applied 2 Smart Smoother IQ filters on top of each other.

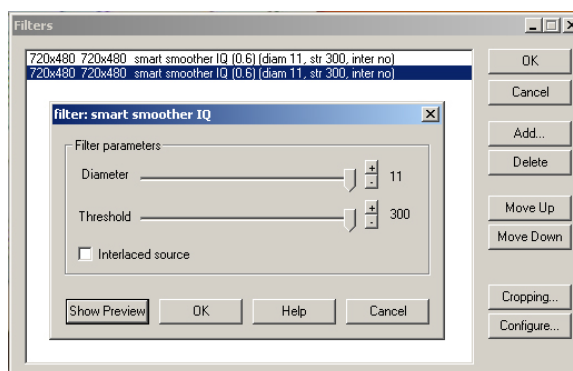


Figure 58: Smart Smoother IQ applied twice, both times with the maximum values.



Figure 59: While it might be hard to see in black and white images, this scene from Sakura Wars is plagued by rainbows.



Figure 60: After two passes with Smart Smoother IQ, our scene has considerably less rainbowing than before.

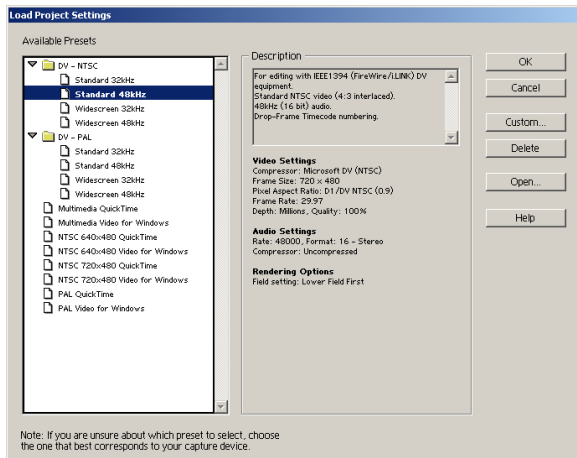


Figure 61: The Load Project Settings dialogue. None of these are good enough for us, so we're going to press the custom button and set our own template.

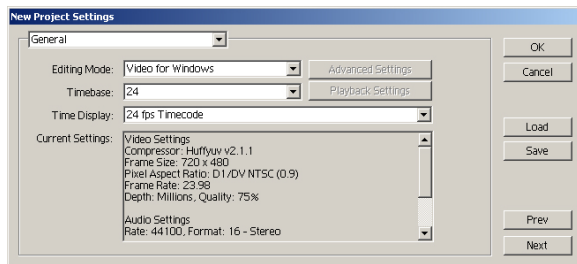


Figure 62: General project settings. These settings affect how Premiere sets up our timeline.

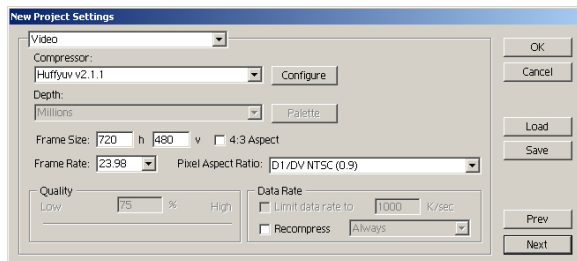


Figure 63: Our project's video settings. It's very important that your project settings, and your output settings, match your clip's settings to maximize editing speed and video quality. The pixel aspect ratio has been changed so Premiere will resize the contents of the preview window to more accurately reflect what the image will look like on an NTSC television.

Setting Up Adobe Premiere 6.01 to Work with Our Ripped Footage

Adobe Premiere 6.0x is dated, but it's way more stable than 6.5. Later builds of Premiere dubbed "Premiere Pro" require WindowsXP to run, and anybody who works with computers on a regular basis can tell you that "WindowsXP Professional" is an oxymoron. Most of this information can be applied to Premiere Pro, the layout and such just might be different.

Open up Premiere 6, and you'll be presented with the Load Project Settings dialogue. If not, go to file/new project. Click the "custom" button on the right.

The first section is the "general" section. Here, we specify "Video for Windows" as the editing mode using the drop down menu. Set the timebase to 24, and the time display to "24 fps Timecode". This won't affect our video any, it simply modifies the timeline so that 1 frame on the timeline equals 1 frame of video. You can use 24fps source and work on a 29.97 fps timeline, but you'll run into synchronization problems.

Click the "next" button on the lower right to bring up the video properties. For the compressor, Choose Huffvuv v2.1.1. Huffvuv only has 1 real quality setting, and that's perfect, so quality and depth grey out. Under frame size, type in 720 h and 480 v. Uncheck the box next to 4:3 aspect if you can't type these numbers in. Don't settle for 710x480. Set the framrate to 23.98 (23.976 rounded up). For pixel aspect ratio, choose D1/DV NTSC(0.9). What this does is resize the video on the fly in the preview window to be sized like it would be on the tv. The edges may look jagged in the preview window, this is fine as it doesn't affect our final output. Uncheck the "recompress" checkbox near the bottom.

Working with Footage

Click the “next” button again to bring you to the audio properties. Here we want to set the audio properties to match the audio that you’ve prepared for your AMV. Music on CDs is recorded at 44100 Hz, so set that as your rate. A higher setting will make your audio sound hollow and mechanical. Set the format for 16-bit stereo, unless your audio is monoaural. CDs have to be in stereo, so for the most part, you’ll never have to change this again as well. For the compressor, select uncompressed, as we don’t want to compress our audio.

Even if your source audio is an mp3, it’s HIGHLY recommended to convert that mp3 to an uncompressed .wav file first. There are a lot of tools to do this, including the disk writer plugin that comes bundled with the free, and popular, Winamp by Nullsoft. Premiere can become unstable when working directly with mp3s, and you will run into sync issues.

Set the interleave to 1 second (this is usually the default) and set the enhanced rate conversion to off. We’re not converting the sampling rate, so there’s no need for Premiere to waste time trying to do so.

Click the ‘next’ button one last time to bring us to the keyframe & rendering settings. Make sure all of these checkboxes are unchecked. Contrary to popular belief, “optimize stills” just messes up your images. Change the preview to “from RAM” meaning once you play a segment of your video, it stores it in RAM. If you play it again, it should play full speed. Huffiyuv is way too large to preview “from disk”. In the fields drop down, select “no fields” as we’ve already deinterlaced our footage.

Go ahead and click OK. We’re all set up and Premiere is ready to accept our extracted clips.

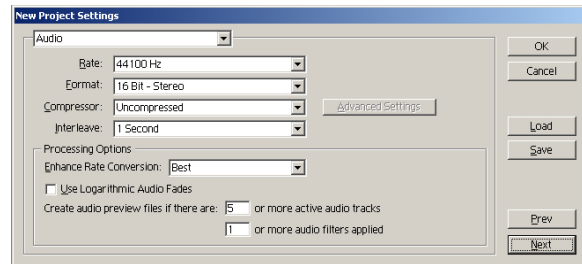


Figure 64: Audio properties in Premiere. Just like with video, we want Premiere’s settings to match the settings on our source, which in this case is a .wav that you have prepared.

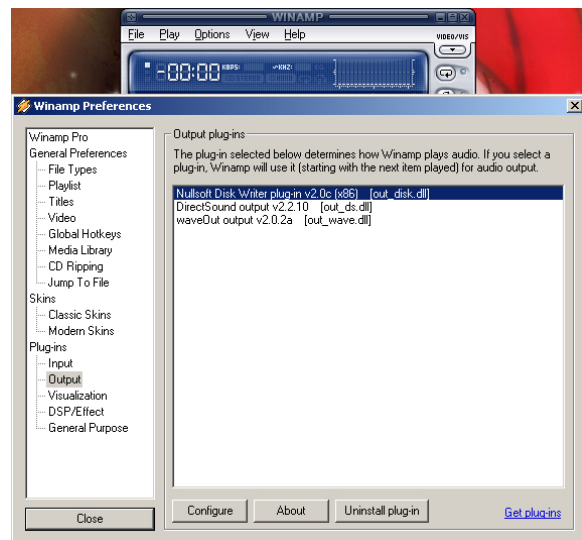


Figure 65: The Nullsoft Disk Writer plugin that comes with Winamp can convert anything being played in Winamp, to a .wav file. Just remember to turn it off when you’re done. For space reasons, we won’t be going over working with audio in this guide.

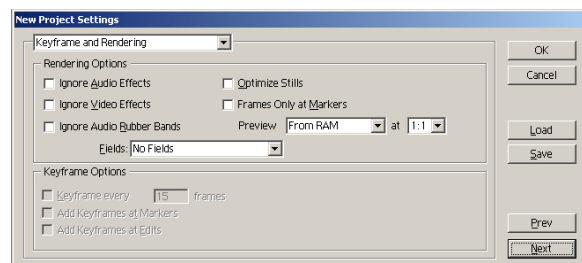


Figure 66: Keyframe and rendering options. It is very important to set the field setting to “no fields”, otherwise Premiere will interlace any effects you add, including camera pans, fades, and lens flares.



Figure 67: We slowed down our deinterlaced footage in Premiere, but by default, it tries to deinterlace the clip again using a very poor method of doing so. Not only are the edges jagged, but they wiggle as the video plays. It is very noticeable on both computer monitors and televisions.

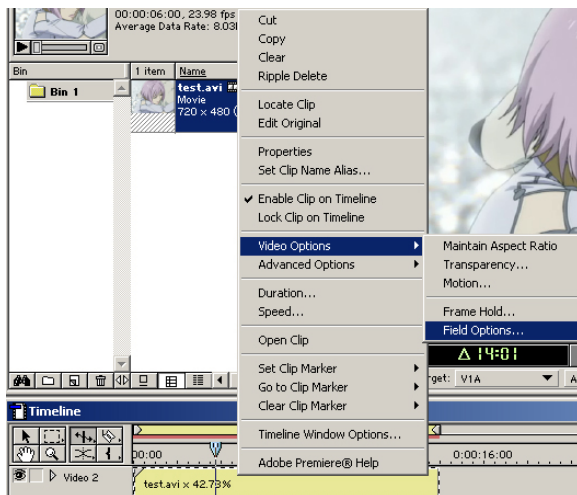


Figure 68: Right click on the clip and select video options/field options.

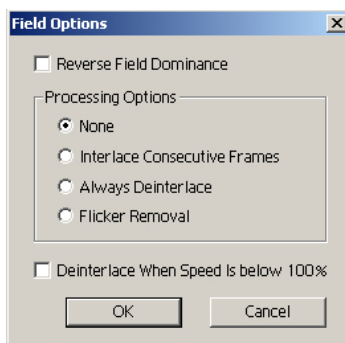


Figure 69: Uncheck “Deinterlace when speed is below 100%”

A Note About Changing the Speed of Our Footage in Premiere

If you slow down your footage in Premiere 6.x, Premiere will automatically try to deinterlace the clip using the absolute worst method possible. If we were working with interlaced footage, maybe we’d want this, but our footage is progressive because we’ve already deinterlaced this. The only way to stop Premiere from having its way with slow clips is to change a certain setting on every clip that we slow down.

Premiere Pro 1.5 does not have this problem if told that we are working with “no fields”.

To fix this, on the timeline, right click on a clip that has been slowed down. Roll over “video options” and a new menu will expand next to it. Select the bottom option labeled “field options”. A new window should pop up. The checkbox at the bottom reads “Deinterlace when speed is below 100%” and is checked. Uncheck it, and hit OK. You will have to do this for every clip you slow down with Premiere 6.x.

Exporting Your AMV from Premiere

Exporting is really easy if you've been following along. Just set all your export settings to match your project settings. The whole trick is to not let Premiere change anything, especially your video size. Premiere resizes with a nearest neighbor filter which works fast, but looks bad, so it's essential for the export to match the project which matches the source.

When you're ready to build your final .avi, go to the file menu, select 'export timeline', and then select 'movie'. Likewise, you can hit CTRL+M. Give your video a name, and click the settings button below it. A new window will pop up with your export settings. The first page is the general options, and is a little different than our project's general options. For file type, select Microsoft AVI. For range, use 'entire project' if you have nothing extra before or after your video, otherwise mark off the area you want on the timeline and use 'work area only' to export only a segment. Check 'export video' and 'export audio', and you can check 'open when finished' as well, if you want the video to open inside of Premiere when you're done.

Click the "next" button and you're presented with the video options. Set this exactly the same as your project's video options. Do the same for the audio settings, and the keyframe and rendering settings. Don't touch anything in the final page named "special processing". There are no enhancements there that we want to use.

When you're done, click OK, and then click save. A progress meter will pop up while your render is in progress. Depending on the complexity of your video and the speed of your machine, it can take minutes, or hours, to export a whole AMV. The final huffyuv .avi file will also be quite large, so check your space ahead of time.

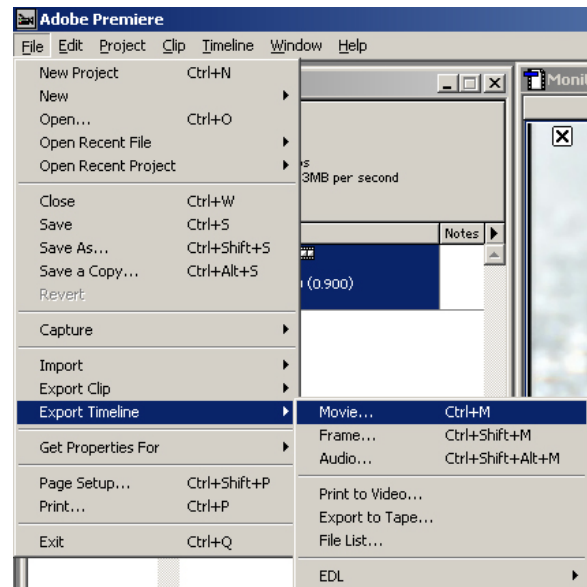


Figure 70: Exporting our final AMV with Premiere 6.x.

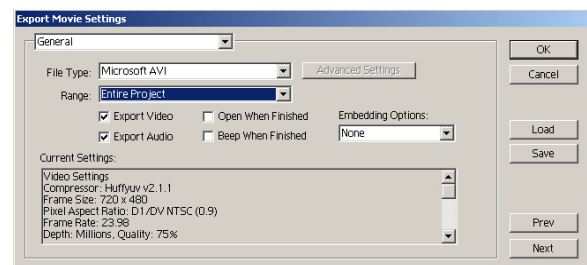


Figure 71: General export settings. We've seen all the other pages of settings before when we first set up our project, this is the only page that's new to us.

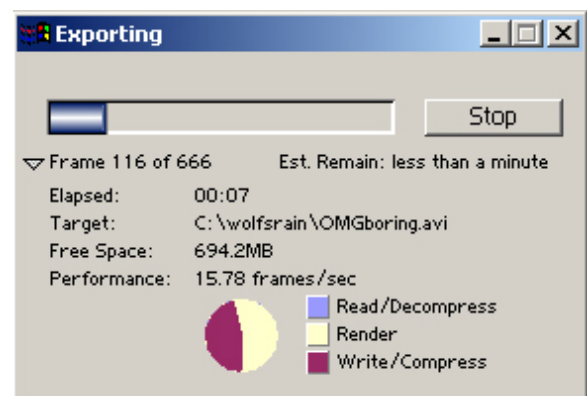


Figure 72: Premiere's progress meter. The meter will slow down considerably every time Premiere hits a clip with an effect on it. In the meantime, you really should go outside, it's good for you...

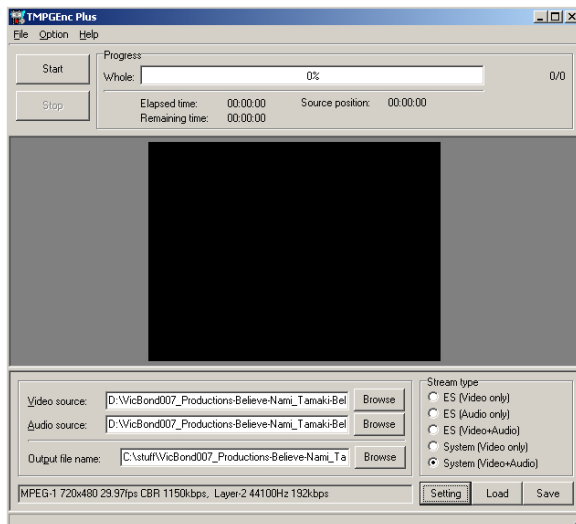


Figure 73: Our video loaded into tmpegenc. Notice how it occupies both the fields for video and audio source. The stream type has also been set to ‘system (Video+Audio)’

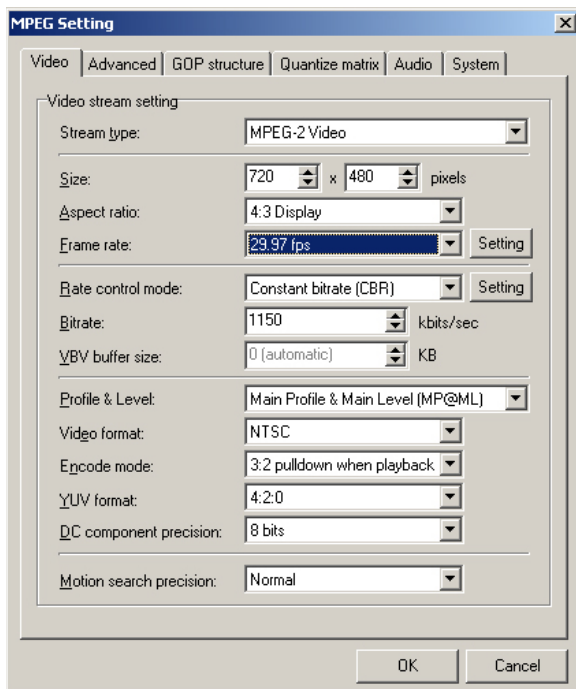


Figure 74: Making our first changes to the video settings. To open up the appropriate options, we have to change the stream type to mpeg2, and the encode mode to 3:2 pulldown with playback, in that order.

Distribution

Making an Mpeg2 That’s Suitable for an AMV Contest Screening

After exporting our final huffyuv avi, you probably notice two things. For starters, the video is huge. Like, 2GB+ huge. Secondly, it plays choppy. What we’re now going to do is use a program called TmpegEnc to compress an mpeg2, suitable for submission to any contest that accepts digital entries (such as those run by Otaku Video).

For starters, download tmpegenc from here: http://www.tmpegenc.net/e_main.html

Install it, and load it up. Since this is a demo version, we only have 14 days in which we can make mpeg2s. After that, we’re limited to mpeg1 until we buy it. No functionality is crippled however. Because there are a lot of settings, I can’t really go into the reasoning behind all of them in this guide, so I’ll let the screenshots do the talking, and explain some of the more questionable decisions.

For starters, load up your footage by clicking the browse button near the bottom, next to “video source”. Find your footage and press OK. The path to your video should appear in the video source box, and the audio source box. If you don’t like the output location, browse for a new one. When you’re ready, click the settings button to the right of these fields.

For starters, set the stream type to MPEG-2 Video. Then go down to “encode mode” and set it to “3:2 pulldown when playback”. This tells the mpeg2 to telecine itself on the fly, when played back. This way it meets NTSC spec and can be displayed properly on televisions and LCD projectors without appearing jerky.

Distribution

Your size should be already set to 720x480, and your aspect ratio should be 4:3 Display. The aspect flag is very important because it tells mpeg2 decoder cards, DVD players, and even windows media player (when in fullscreen mode) how to stretch your video to be displayed properly. Even if your video is 16:9, if you followed the letterboxing tutorial and added borders yourself, you should still flag your video as 4:3.

For frame rate, change it to 23.976. Because we selected 3:2 Pulldown with Playback, it will also tell us that equates to 29.97fps internal, meaning after telecine, the framerate will be 29.97.

For rate control, select manual VBR, and then hit the setting button. for maximum bitrate, enter 9350 kbits/sec. DVD, and some decoder cards, max out at 9800 kbits/sec, and that includes audio, as well as a little extra overhead for the system stream. 9350 is as close as you can get while still keeping it safe. For minimum bitrate, put 0, meaning if tmpeg can draw the image perfectly without using as much as 9350kbits/sec, it will use as little bitrate as possible, but still take as much as it needs (up to 9350). For P picture spoilage and B picture spoilage, enter 0 for both of the values. Spoilage is how much you'll allow tmpeg to degrade the image in the interest of file size. We don't care about file size, so we tell it not to spoil.

Hit OK, and set the VBV buffer rate to 224. This is the number required if you ever wanted to dump your video to DVD. Setting it to auto could produce a slightly better picture, but it would not be DVD ready.

The other two fields you have to change are DC Component precision (set to 10 for the most accuracy) and Motion Search Precision (set to highest quality: very slow).

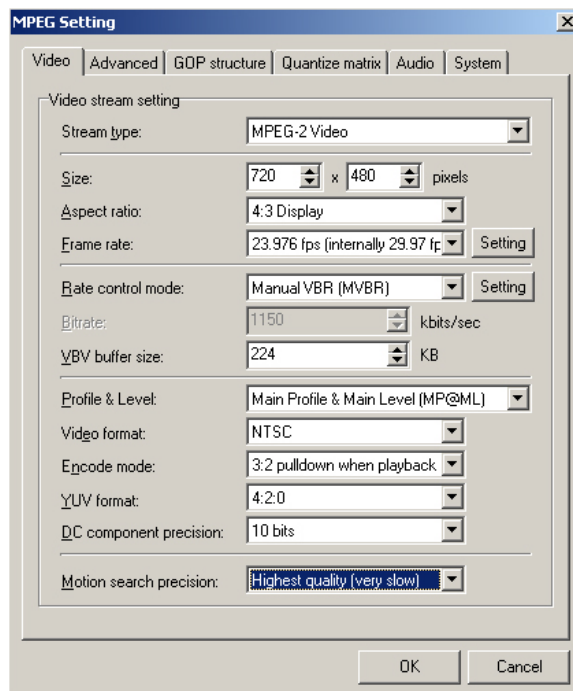


Figure 75: Recommended video settings to distribution to a convention. So far, our video stream is DVD compliant, and of substantially high quality.

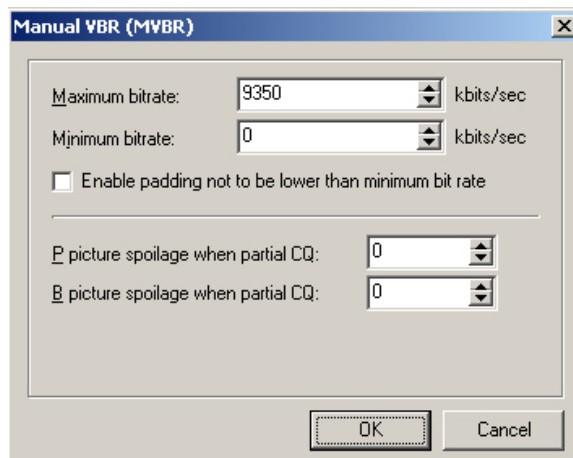


Figure 76: Recommended bitrate for convention distribution.

Distribution

The next tab is the audio tab. Here we want to set our stream type to Mpeg-1 Audio Layer II. This stream will not be DVD compliant because audio on an NTSC DVD can only be uncompressed (no room given our large video bitrate) or ac3 (Dolby Digital requires a much heftier licensing fee than our \$30 registration for tmpeg). This should not be a problem though because any convention that is mastering a DVD, should have enough sense to know how to extract and re-encode your audio.

Set the sampling frequency to 44100 to match our current sampling frequency. Set the channel mode to “stereo” to ensure maximum stereo separation. Joint stereo tends to blur the left and right channels together in the interest of saving file size. Set the bitrate to 384, which will produce an almost lossless encoding of our audio track. Finally, tick off the “error protection” checkbox to ensure a steady, in-sync stream.

In the system tab there is nothing that we have to change. The program type should have automatically set itself to “MPEG-2 Program (VBR)”. You can check “insert comment” and type in something to hide in the stream if you so desire. Certain media players will be able to display this.

Click OK, and then hit the start button located in the top left corner of the original window. Quality mpeg encoding is VERY slow but in the end, you’ll get a very clear video that takes up very little space compared to the huffyuv avi that we rendered out from Premiere.

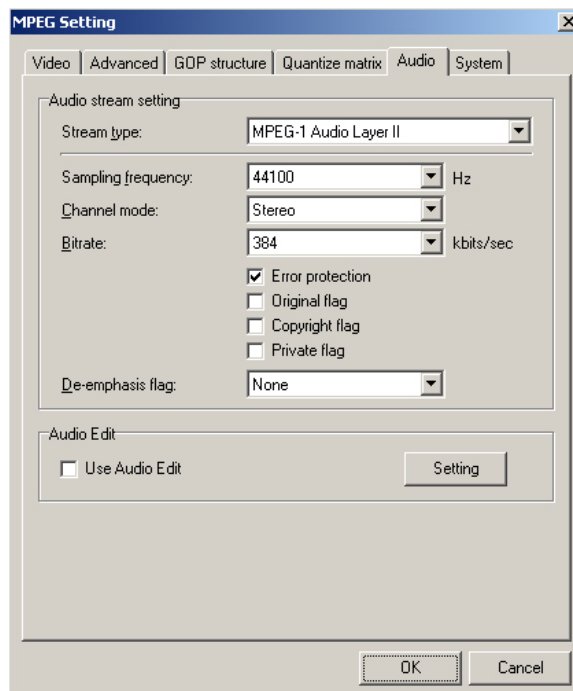


Figure 80: The audio settings tab in tmpegenc, and the recommended settings.

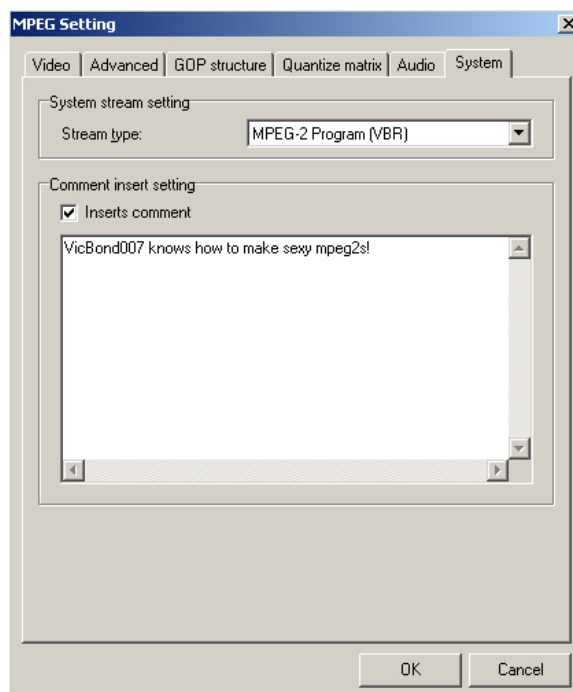


Figure 81: Tmpeg’s system tab. We shouldn’t have to change anything here.

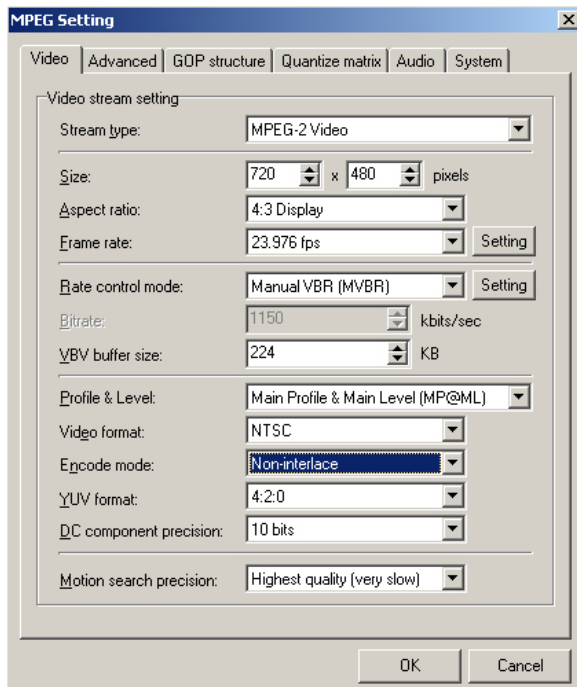


Figure 82: Making a non-interlaced encoding with tmpeg. Change all of the settings in all the other tabs the same way as mentioned earlier.

Creating a Progressive Scan MPEG2 for Conventions without MPEG2 Hardware, or for Personal Use.

While our 29.97FPS mpeg2 works great on computers with mpeg2 tv decoders (Such as Sigma Designs' Netstream2000) and set-top DVD players, it has one problem. We had to telecine our video so it displayed properly on these mediums, and so we added interlacing. Some conventions (check before you submit) run their AMV screenings by loading up all the digital entries into a playlist, and playing them in media player via their computer's desktop tv-out adapter or laptop's VGA-output. For these situations, and for your own personal archive, you may want a copy of your video that maintains the original 23.976fps and deinterlaced nature.

Set everything up as mentioned on the previous pages, however in the "video" tab, set the encode mode to "non-interlace". The frame rate should now simply read "23.976". There is nothing else that is different from our original configuration. Click start, encode-away, and show off your work! Remember, if you're playing this video in Windows Media Player, be sure to run it in fullscreen mode so that it scales to the proper aspect ratio!

CONGRATULATIONS! YOU'RE DONE!