How to downsample SFD files

by TuxTheWise

Introduction

SFD is the video format most used in Dreamcast. It consists in usual MPEG 1 video multiplexed with (except for header change) ADX audio.

Multiplexed means that video and audio were separated into blocks and mixed together into a single file, what is used in practically every movie format. The multiplex system of SFD is the same used in MPEG 1 (and 2?) movies.

Thus to demultiplex a movie means to get the movie file and reconstruct the audio and video in separated files again.

Note that this guide has much more information than you need, so it's up to you to filter it.

There are examples at the end of this document, so check them if anything gets too obscure.

About the programs

- The programs will have indication about where to get them when they're mentioned for the first time.
- Katana SDK means that the program is included as tool in Katana SDK (lol?). Nowadays you can get it in a lot of places (~~including snesorama~~). I also included a copy on this pack.
- AutoIt Scripts need AutoIt Script (lol?) to run. You can get the interpreter here: http://www.autoitscript.com/. If I provide a script, just save it with .au3 extension and double-click it (if your instalation has associated the .au3 extension to run).

Step 1: Demultiplexing a SFD

Any MPEG demultiplexer should be able to handle SFD files.

You can go with TMPGEnc (there is a free version in http://www.tmpgenc.net/en/download.html). Select File -> MPEG Tools... -> De-multiplex tab (Simple De-multiplex won't work with SFD) -> Browse. Select your SFD file. There should be an audio_stream and a video_stream (ignore the others). Double-click each one of streams and save them whenever you want. Use .sfa extension for audio and .mpv extension for video.

You can also use demux.exe (it's somewhere around this pack). I like this one because it's command line tool, what makes batch demultiplexing very easy. Use this syntax:

```
demux.exe video.sfd output
```

It should create "output_c0.m2v" (rename it to .mpv) and "output_e0.m2a" (rename it to .sfa).

If you want to batch multiplex, put demux.exe and demux.au3 (both can be found around on this pack) in your SFD directory, execute the script and it will create a "montar.bat". Execute the batch and it will demultiplex every SFD of a directory. Very useful, I use it all the time.

Intermission: DC Movie Maker - The easy and crap way

DC Movie Maker is the official SFD tool included in the Katana SDK. It can encode videos and audio (of a variety of formats) and multiplex them into a SFD file.

I don't recommend using it for encoding media because it only encodes MPEG using CBR (constant bitrate). VBR (variable bitrate) offers more quality with reduced size, because it adapts the bitrate according to the scene you're encoding. As an example, you'll get a much smaller file encoding a credits video using VBR than using CBR, because there are a lot of steady areas.

Additionally, I believe DC Movie Maker MPEG-1 encoder doesn't have a very good quality. I haven't tested it too much though.

About audio encoding, I have nothing to say about it.

DC Movie Maker may be useful for multiplexing audio and video though.

If you're thinking about using it, I have some commentaries. Note that if you don't want to use it you can just skip to the next chapter.

1) If you use a .mpv file as video input, DC Movie Maker will think it's not a ready MPEG video stream and it will re-encode it (even if it's ready!). Use it if you want to downsample a video.
2) If you use a .m1v file as video input, DC Movie Maker will think it's a ready MPEG video stream and won't change it, it will only multiplex it into a SFD file.
3) If you use a .sfa file as audio input, DC Movie Maker will understand that it's a ready audio stream for SFD and won't change it, it will only multiplex it into a SFD

file.
4) If you use other format (like .wav) as audio input, DC Movie Maker will encode it and multiplex it into a SFD file.

Step 2: Working with video
To downsample the video, you'll need an encoder. Like I said, you could use DC Movie Maker, but you'll get very poor results. I recommend two encoders:

- TMPGEnc. It's freeware, it supports VBR encoding, and it outputs good results.

To encode a video into a lower quality, at the bottom of screen select your input.mpv (.mpv is not shown as a supported format, but load it anyway). Select "ES (Video Only)" as Stream Format, and click the Setting button right below it. Select "2-pass VBR(VBR)" as Rate control mode, and click the Setting button at its side. Here, adjust the bitrates values (discussion about it will be made later), confirm the options. Adjust the output file if needed and start the encoding with the giant Start button at the top. Wait until it ends and you'll have your downsampled video.

- Mainconcept MPEG Encoder (developer website: http://www.mainconcept.com). It's a commercial, very expensive codec. In my option, it presents the best results, noticeable most in dark scenes. It's also very fast, with native support for special instructions (SSE2 and above) and multi-core. You can't go wrong wasting all these hundred dollars.

Since only a few people have access to this software, I'm not telling you how to use it. It's pretty much the same of TMPGEnc, so you shouldn't have problems.

Note that batch support in both of these softwares aren't that good if you need to adjust space and try a lot of different bitrates. I don't have an easy solution to give you.

Step 3 (optional): Downsampling Audio
Before going on with that, note that audio is a very important part of video, and it only uses a small amount of data compared to video. It means that the general rule is "don't touch the audio" unless you have a very limited space.

For example, imagine a SFD with video encoded in 4800kbps CBR and audio being 44khz stereo (397kbps). Well, audio in this case uses only ~7.6% of the space in the file. If you mono the audio (198kbps), you'll lose a lot of quality in audio but your video will only be ~3,8% smaller.

Anyway, let's teach you how to work with audio.

Decoding .sfa

Rename it to .adx, open it with foobar2000 (freeware, download it in http://www.foobar2000.org/). Right-click the file in playlist, click Convert and use the appropriate format (WAV).

Note: .sfa and .adx have differences in header, but foobar2000 can handle it.

Note: If foobar2000 doesn't recognize ADX, you'll have to install the correct plugin for it (available on foobar2000's website).

Note: There are a lot of crap old decoders for ADX, so be sure to use a good one (like foobar2000). Probably you'll also get good results with ffmpeg (free, pre-compiled Windows version here: http://ffmpeg.arrozcru.org/autobuilds/).

Modifying audio

You can convert the decoded audio to mono and/or downsample it to save space. I like GoldWave (commercial, simple and useful, official site website here: http://www.goldwave.com/), but you can also try Audacity (free, available here: http://audacity.sourceforge.net/)

I'm not teaching you how to use these programs, you should be able to figure it out by yourself.

Encoding Audio Back

Use "adxencd.exe" (Katana SDK) to encode your .wav into a .adx file:

```
adxencd.exe input.wav
```

The parameters (number of channels and frequency) are given by the .wav used as source.

Note: You'll find a tool to batch encode ADX files somewhere around this pack.

Converting ADX to SFA

Use ADXConverter.exe (included in Darkfalz's DC Tools, it can be found somewhere around this pack) to convert the ADX files to SFA. It has native support for batch

conversion.

Final step: Multiplexing files back to a SFD file

You can use DC Movie Maker to multiplex the files. If you want to use it, check its chapter above.

I like using sfdmux.exe (hard to find wrapper for official sfdmux.dll, made by Zench. I found it here: http://forum.xentax.com/viewtopic.php?f=18&t=3084 but you can found a copy around this pack. It's called by command line so you can easily batch multiplex files.

Well, at this point, you must have your video .m1v (modified or not) and your audio .sfa (modified or not). To multiplex then into a SFD use:

```
sfdmux.exe -V=video.m1v -A=audio.sfa -S=output.sfd
```

and you're done.

You can use criar.au3 (included around this pack) to automatize the process. It will create a "montar.bat" to multiplex all the .m1v of a directory, assuming that the .sfa has the same base name.

Extra: About Downsampling and Bitrates

The main reason for downsampling SFD is to save space. But there are situations when you need to downsample movies to make them play right.

Videos at the "slowest" part of the disc cannot be higher than ~3600kbps CBR, else reader won't read it fast enough (GDs are read faster than CDs). This bitrate limit is higher at the "fastest" area of the disc (I remember testing ~5000kbps CBR without problem). Some games (e.g: Spawn, Star Wars Jedi Power Battles) have videos encoded in a very high bitrate (~9000kbps), and there is no way to play it right on a CD. So they'll need to be downsampled even if you have enough space in the disc.

Before choosing your new bitrate, don't forget to check the one of the original video. You can see it after select a SFD or M1V or MPV at the De-multiplex tab of TMPGEnc.

Now about minimum and maximum bitrates choices. For minimum, you can always use 0~300. For maximum, just use your good sense. For example, if my average

bitrate is 3600kbps, I'd choose a maximum bitrate of 4000~4400kbps.

Oh, and don't forget that re-encoding always make you lose quality, even if you use bitrates higher than the original ones.

About dimension reducing

Resizing your video is a bad idea, unless you need to use an incredible low bitrate. General advice: don't use it. Video may lose too much quality, it may lose aspect ratio, the game may not play it right, etc.

I've never tried this, at least at the time that this document was written.

Math prediction

VBR encoding may be a pain in the ass when you want your videos to have a determined size. Like the name says, it uses a variable bitrate, so the final file size can't be predicted with precision like you could do with CBR. It will force you to re-encode your videos a lot of times until you find a bitrate set that gives you the size that you want.

If you use a too high bitrate, it won't fit on a CD. If you use a too low bitrate, quality will be worse than it could be. What to do then?

Imagine that you need a .m1v video to have 20mb. You use an average bitrate of 1000kbps and get a result of 10mb. Well, using the good sense, if you double the average bitrate (1000kbps -> 2000kbps) you should also double the video size (10mb -> 20mb) and solve your problem. Result won't be exactly the double because of VBR variations, but the double bitrate is your best next shot to try. But what if your numbers are too complicated? And you actually want to have the size of the SFD file, not of the single video stream. So how to consider all that to find your next bitrate try? You can manually guess the bitrates, but believe me, it will take dozens of encoding.

Well, this chapter will show you how to estimate the next bitrate try based on your last try, so you'll reduce the number of times you need to encode until find the right size. Don't worry if you don't fully understand it now, it will be clearer in third example at the final of this document.

Facts:

1) The space your final video must have is T.
2) The size of the SFD of the last encoding with bitrate B* is S*.
3) The size of your .sfa is A.
4) The next bitrate you need to try is B (that's what you want to estimate).
5) When multiplexing a file, the final size will be ~1.0157 * (Size of video + Size of Audio). This is an experimental value and it works very fine.

(S*/1.0157) - A = Size of video stream (should be obvious if you think a little).

Using simple proportion:

B* / B = [(S*/1.0157) - A] / [(T/1.0157) - A]

And with some algebra we get:

$$B = \frac{B^*(T - kA)}{S^* - kA}$$

where:

k = 1.0157
T = Size that the video must have.
A = .sfa (audio stream) file size.
S* = Size obtained on the last encoding.
B* = Bitrate used on the last encoding.

As I told you, things will get clearer at the correct example later.

Note that you can use the formula for a bunch of files, just replace the parameters for the sum of them (except for bitrate of course). Obviously, you'll have to use the same bitrate for all the videos in order to do that.

Example 1: Downsampling Sonic Adventure Opening without change the audio

File information:

SAH448_6.SFD
49491968 bytes
Video bitrate: 3200kbps CBR

Work Directory: H:\Tutorial SFD\

Plan: Downsample it to an 2000kbps (average) bitrate using VBR.

1)

```
demux.exe SAH448_6.SFD lol
rename lol_e0.m2v SAH448_6.MPV
rename lol_c0.m2a SAH448_6.SFA
```

Result:

SAH448_6.MPV - 43461666 bytes
SAH448_6.SFA - 5299268 bytes

2)

In TMPGEnc:

- "H:\Tutorial SFD\SAH448_6.MPV" as video source.
- "H:\Tutorial SFD\SAH448_6.m1v" as output filename.
- Select "ES (Video Only)".
- Setting -> Select "2-pass VBR(VBR)" -> Setting.
- Average bitrate = 2000kbits/sec; Maximum bitrate = 2600kbits/sec; Minimum bitrate = 0.
- OK -> OK -> Start.

Result:

SAH448_6.m1v - 27126950 bytes.

3)

```
sfdmux.exe -V=SAH448_6.m1v -A=SAH448_6.sfa -S=SAH448_6.SFD
```

Result:

SAH448_6.SFD - 33144832 bytes (~33% smaller).

You can test it, it will play correctly in the game.

Example 2: Downsampling Sonic Adventure Opening, but now monoizing the audio

Same file information and working directory.

Plan: Downsample the video to an 2000kbps (average) bitrate using VBR.Make audio mono.

Steps 1 and 2 are exactly the same of the last example.

3) `rename SAH448_6.SFA SAH448_6.ADX`

4) Open SAH448_6.ADX on foobar2000 (it should play). Right-click playlist entry -> Convert -> ... -> Output format = wAV -> OK.

5) Open SAH448_6.WAV in GoldWave. File -> Save As -> Select PCM signed 16bit, mono in Attributes -> Save (overwrite previous .wav).

6) `adxencd.exe SAH448_6.wav`

Result:

SAH448_6.adx - 2649294 bytes

7) Open ADXConverter.exe. "H:\tutorial SFD\" in the main field -> Mark ADX to SFA -> Mark Delete ADX -> Convert.

Result:

SAH448_6.sfa - 2649546 bytes

8) Do step 3 of last example.

Result:

SAH448_6.SFD - 30455808 bytes (~38% smaller).

You can test it, it will play correctly in the game.

Note: We have monoized the audio. It means that we lost a lot of quality in audio but gained only ~5% in size compared to the video with the downsampled audio. That's why you should avoid changing the audio.

Example 3: Math prediction

Note: Everytime I encode the video again, it's clear that I'm using the original video as source! Don't be dumb and use your previous result as source.

Imagine the result of the last example:

SAH448_6.SFD - 30455808 bytes
Average video bitrate: 2000kbps.

But you actually had 40000000 bytes free, so you can use a higher bitrate to improve the quality. What bitrate will you use now? Let's use our prediction formula (in fact my formula, but that's cool):

Using:

k = 1.0157
B* = 2000
T = 40000000
A = 2649546 (size of the monoized .sfa)
S* = 30455808

and applying on the formula, we get B ~ 2688 kbps.

Encoding the video now with 0/2688/3000 and multiplexing files we get:

SAH448_6.SFD - 39890944 bytes

That is incredible close to the result we wanted (only 0.2% smaller). We could stop here, but let's suppose that the result was not so close as we wanted. We could do the same process again to find our next bitrate try. Using:

k = 1.0157
B* = 2688
T = 40000000
A = 2649546 (size of the monoized .sfa)
S* = 39890944

We'll get B ~ 2695. If we encode again using 0/2695/3000 and multiplex we get:

SAH448_6.SFD - 39979008 bytes

Even closer! Cool right?

Note that this formula will never converge to the exact result because of VBR variations. It should however be floating around the exact value.

Also note that the maximum bitrate actually affects the final size of file, so you should maintain it fix once you gets close to the result you want.

This formula converges pretty fast and gives you very useful tips when you have a restricted space for video. Use it!

Blue Stinger Anomally

This "anomally" also happens in Flag To Flag, and it may happen in other old games (all Dreamcast games are old now, but you got the idea) that use SFD videos. The old SFD multiplexed format uses additional header space and you need special attention to work with them.

If you get an error message before demultiplexing it, it's probably the case.

Before demultiplexing:

Find the first "00 00 01 BA" ocorrence (I think it's always in the 0x2000 offset) using your favorite Hex Editor (like Free Hex Editor Neo, freeware, available here: http://www.hhdsoftware.com/Products/home/hex-editor-free.html). Remove everything above it (yeah, rip them off), and save the changes.

Now you should be able to demultiplex the streams without problems.

After multiplexing it back:

Use SFD-Fixer (included somewhere on this pack) the program should be self explainable. In the "Original video" field select the original file (lol?), not the one that you modified in the hex editor.

After that the video should play fine in the game.

Sonic Adventure 2 Anomally

Videos in Sonic Adventure 2 have multiple audio streams (Japanese and English audio). Actually it's the only game I've ever seen with this characteristic.

You'll be able to demultiplex the audio fine, just remember that you'll get two audio streams.

To multiplex a SFD with the two audio streams back, use:

```
sfdmux.exe -V=video.m1v -A=audio1.sfa -A=audio2.sfa -S=output.sfd
```

Note that parameters order in sfdmux.exe matters!

Also note that you can't multiplex multiple audio streams using DC Movie Maker.

If you don't add the two audio streams, the game will freeze when it tries to play the video with English audio.

Final words

Wow, it got longer than expected. I hope it can be useful for someone.

Let a comment if you have questions, just be sure to read the full guide before asking stuff. I don't like to answer questions quoting parts of my own guide.