

ACTIMAGINE

Mobiclip Middleware for NINTENDO DS

Quick start guide

Introduction

The Mobiclip Middleware for NINTENDO DS is divided into different components:

- AviToMobiclipDs conversion tool.
- Video resources.
- Four code samples.
- Mobiclip library and prototype files.

Each of these components will be described below.

History

*Version 1.0.1 (March, 14th 2008)

.Sample code modifications:

- define 4KB of stack to free up DTCM space
- use OS_GetDTCMArenaHi and OS_GetDTCMArenaLo to compute available DTCM space.
- added OS_CheckStack call after MO_OpenMovie call to check that stack is not overwritten by Mo_OpenMovie function.

.Html documentation updated:

- added information about the need of specific stack allocation in lcf files to free up DTCM space for Mobiclip usage).
- all references to 'SDRAM memory' replaced by 'main memory'.

* Version 1.0 (March, 3rd 2008)

.Initial release

Mobiclip SDK For NINTENDO DS general features

Video playback features :

- During single-screen display, up to 30 frames/second for full screen (256x192) playback
- During two-screen display, up to 15 frames/second for full screen (256x192) playback on both screens
- Bit rate ranges from 100 to 700 Kbps/screen

Audio playback features

- Supports both mono and stereo audio
- Frequency ranges from 8 to 48 KHz
- Data rates:
 - For mono tracks, the rate ranges from 0.6 to 1.25 bits/sample
 - For stereo tracks, the rate ranges from 1.2 to 2.5 bits/sample

Mobiclip Video Codec for Windows

About protection

The Mobiclip Video Codec for Windows is protected by the StarForce protection system, which provides security by only allowing the codec to be used on authorized computers and by allowing the codec to only be used for a limited period of time.

Codec installation

Please read the documentation “Nintendo - MobiclipSDK - Starforce protection - User Guide.doc”

This documentation describes both the process for installing the codec and its activation.

Video and audio processing

Video editing tool

We recommend using the freeware VirtualDub as a video tool to process your videos. You can download it at <http://www.virtualdub.org/>.

You can also use “off the shelf” products from other software vendors.

Compatibility issues

Some applications do not conform to the Video For Windows 1.1 standard (VFW 1.1) because they do not provide the codec with the frames per second information or the total number of frames information.

The codec detects such an incompatible application, and displays a popup that tells the user what information the codec lacks.

Known incompatible applications:

- Adobe After Effects CS3: it doesn't give the Frames Per Second information nor the Number of Frames information. This encoding mode is the only known working mode:
 - o Single Pass – Quality
- Adobe Premiere Pro CS3, this product doesn't give the Number of Frames information. These encoding modes are the only known working modes:
 - o Single Pass – Quality
 - o Multipass – First Pass / Target Bitrate (kbps)
 - o Multipass – Next Pass / Target Bitrate (kbps)

Video Preparation and Processing

Video preparation and processing are very important. They will contribute to 50% of the final video quality, the remaining 50% coming from the video codec itself.

Rule #1: Use the best video source you can get

Rule #2: Resolution of the video sources should be big, at least 640x480

Many processes (like contrast/gamma) will benefit from the size of the video and create errors (after the final downsize), because they are applied to more pixels beforehand.

Rule #3: Delay the final downsizing to DS resolution as late as possible in your processing pipeline

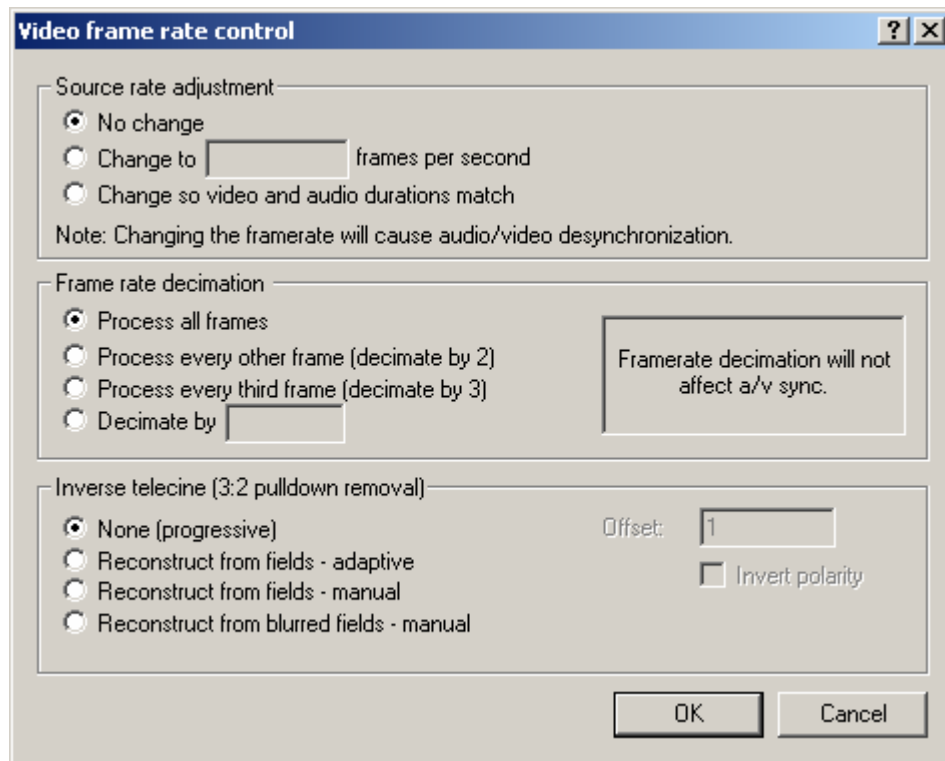
Rule #4: Source video should be uncompressed or not compressed much

Rule #5: Be careful of the source video frame rate

Typical videos captured from TV are 30 fps. However, the original frame rate (cinema) is often 24 fps. In that case you should convert the movie back to 24 fps with an “inverse telecine” method to ensure better quality. This kind of method is available in VirtualDub in the *Video/Frame rate* menu.

Rule #6: When lowering frame rate, only use “decimation” methods

See the *Video/Frame rate* menu in VirtualDub.



Rule #7: Do your best to minimize any changes in the aspect ratio of the source and final video (in general 10% difference is ok)

Your source videos will probably have an aspect ratio of 4/3 (TV), 16/9 (cinema) or 2.35 (cinema also). DS aspect ratio is 4/3 (256/192). To convert from source aspect ratio to DS aspect ratio, you will have to crop your video (remove a part of the image) or add black bars.

Rule #8: Downsize using a “Precise bicubic” (or better) method

You can choose the target resolution of the videos with the following restrictions:

- Final resolution must not exceed 256x192 for DS.
- Width and height of the target resolution must be multiples of 16.

Again, do not forget to preserve the original aspect ratio.

Rule #9: When storing intermediate movies, use a lossless video codec

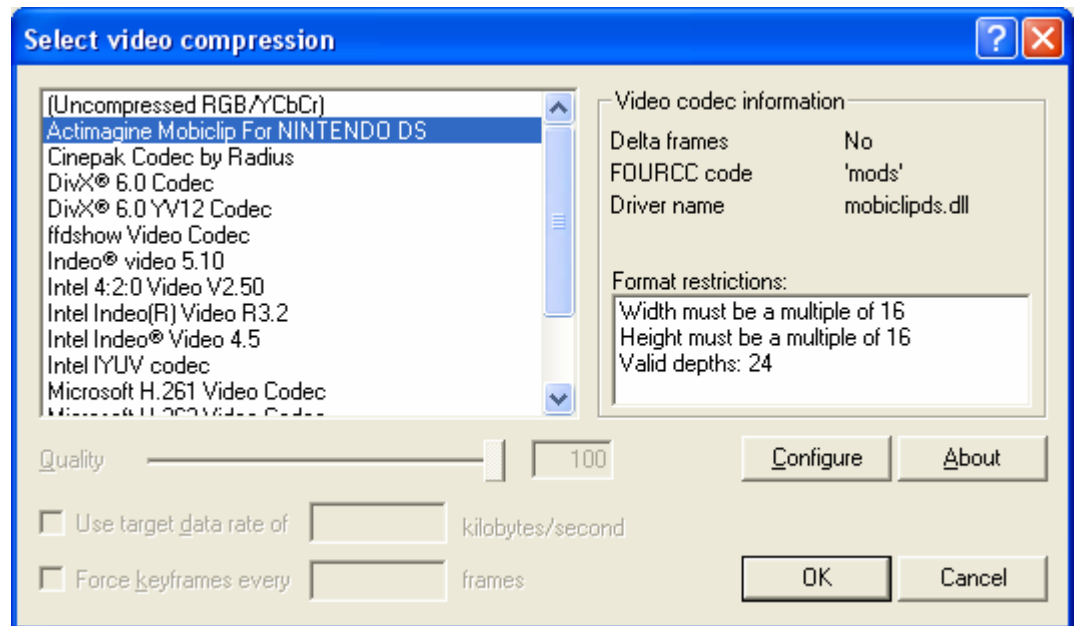
We recommend using Huffiyuv for storing intermediate files.

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

Video Compression

Once the video is prepared, compressing it is straightforward.

1 - In VirtualDub, open the *Video* → *Compression* menu and choose **Actimagine Mobiclip For NINTENDO DS**



2 – Click on the *Configure* button

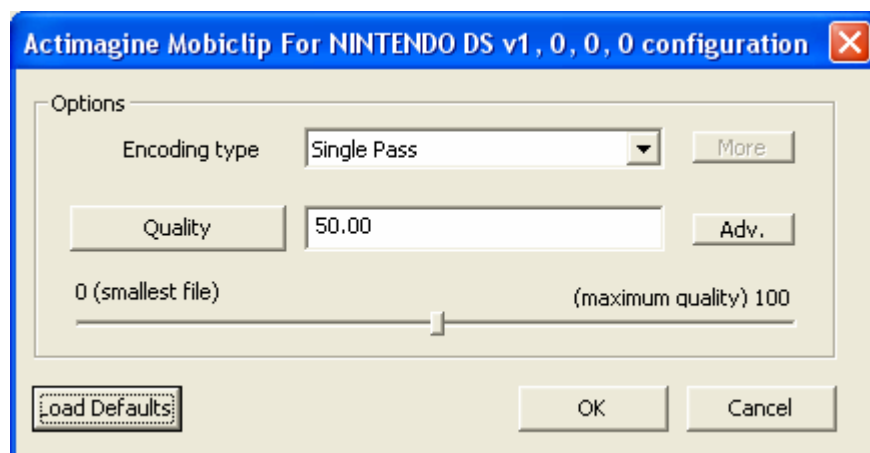
3- Set the parameter values.

You will find below a complete description of the configuration dialog.

4- Close the dialog boxes and launch the compression process.

Mobiclip Video Codec for Windows options description

The following dialog box appears when the ‘Configure’ button is pressed:



Encoding Type

There are three different choices:

- . **Single Pass**

Single pass video encoding type, you can choose between **Quality**, **Target bitrate** or **Target size** methods (see below for a description of the *Quality* button).

The **More** button is not active for this encoding type.

- . **Multipass, First Pass**

The first pass of a multipass video encoding type, you can choose between **Target bitrate** or **Target size** methods (see below for a description of the *Quality* button).

- . **Multipass, Next Pass**

The successive passes of a multipass video encoding type following the first pass. This can be executed any number of times to better target the desired bitrate or size. It should be executed at least once to have a usable video target, and if you call it twice you will already be very close to the targeted bitrate or size. You can choose between **Target bitrate** or **Target size** methods (see below for a description of the *Quality* button) but you should not change the method between the first pass and the next ones or else you may get invalid results..

The **Adv.** (Advanced) button is not active for this encoding type.

Quality Button

It selects the method used for encoding.

If you press it, the name and the meaning of the button will change.

There are three choices possible:

- . **Quality method**

The video encoder targets the quality given by the parameter, and this is the default method. Valid values are in the range 0 to 100, and the default value is 50. You can type the value directly or you can use the slider bar. This method is only available in a **Single Pass** encoding type.

- . **Target bitrate method**

The video encoder targets the average bitrate given by the parameter. Valid values are given in kilobits per seconds and in the range 10Kbps to 1000Kbps, and the default value is 250Kbps. You can type the value directly or you can use the slider bar.

Although this method is available in a **Single Pass** encoding type, it is not the recommended way to use it. To achieve the best video quality possible you should use it in a **Multipass** encoding type.

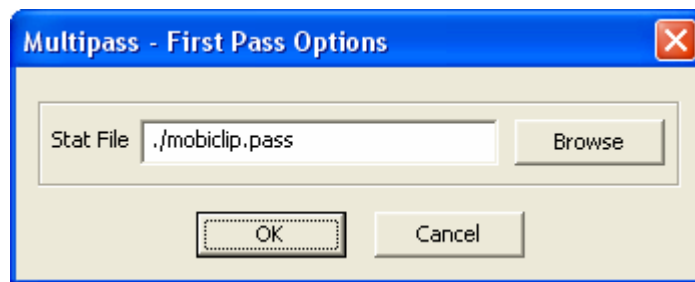
- . **Target size method**

The video encoder targets the size given by the parameter. Valid values are given in Kilobytes, and the default value is 128000 KBytes. Although this method is available in a **Single Pass** encoding type, it is not the recommended way to use it. To achieve the best video quality possible you should use it in a **Multipass** encoding type.

More button

It is only available in **Multipass** encoding types.

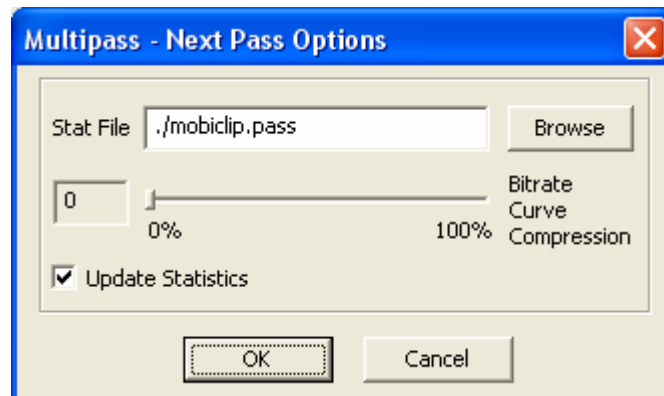
When **Multipass, First Pass** encoding type is selected, the following dialog box appears when the button is pressed:



Stat File

This lets you configure the location and the name of the file which will hold the statistics used for multi-pass encoding. The default file is './mobiclip.pass'.

When **Multipass, Next Pass** encoding type is selected, the following dialog box appears when the button is pressed



Stat File

This lets you configure the location and the name of the file which will hold the statistics used for multi-pass encoding. The default file is './mobiclip.pass'.

This lets you configure the location and the name of the file which has held the statistics during the first pass. The default file is './mobiclip.pass'.

Update Statistics

Check this box if you want the encoding statistics to be updated during the next pass, and it is checked by default. This option is useful if you want to encode a movie using the **Multipass, Next Pass** encoding type more than one time.

Bitrate Curve Compression

!!! WARNING: don't modify this parameter except if you are an expert user !!!

During a multi-pass compression, the codec is trying to achieve a constant quality doing variable bit-rate.

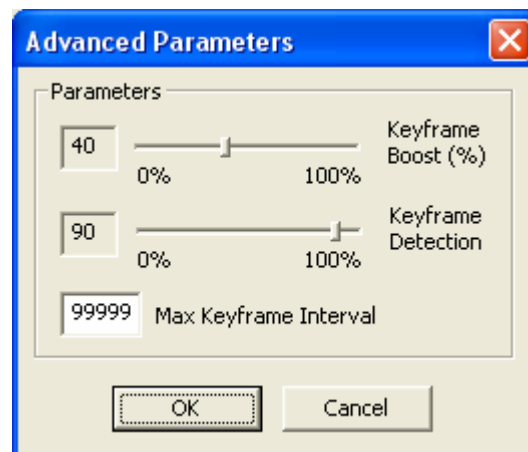
It means that for a given average bit-rate, the local bit-rate at a given playback time can be very high.

As video unpack time is proportional to video bit-rate, a solution to avoid sound buffer underflow during playback is to limit the bit-rate variability.

This parameter allows flattening the bit-rate curve, default value is 0 which means no alteration of the bit-rate curve.

Adv. button

It is only available in **Single Pass** and **Multipass, First Pass** encoding types. The following dialog box appears when the button is pressed:



Keyframe Boost

!!! WARNING: don't modify this parameter except if you are an expert user !!!

This parameter is used to boost the quality of the encoded keyframes of the videos. This often results in a quality increase of whole sections of a video, and its default value is 40.

Keyframe Detection

This is a tuning parameter for scene change detection.

The default value is 90, which means that if more than 90% of the image has changed it will be flagged as a keyframe.

If you have static parts in your video images like permanent black band, you should lower this value. For example if there is only 70% of your images which is changing (30% are permanently black) you should put $70\% * 90\% = 63\%$ as the parameter.

Max Keyframe Interval

This is the maximum interval where the video encoder is forced to put keyframes in your video. The default value is 99999, which means that the video encoder will automatically decide to put a keyframe in your videos (for instance on every cut scene). Putting 0 or 1 will result in a video composed only of keyframes.

Sound Preparation

- Audio format in the final AVI file must be **uncompressed 16 bit mono or stereo**.
- Audio should be **normalized** at 100% so it can use the whole dynamic range of the DS.

AviToMobiclipDs conversion program

AviToMobiclipDs.exe is a console program used to convert an input avi file containing Mobiclip compressed video frames to a binary file suitable for the Mobiclip libraries.

It is located in the directory *Mobiclip/tools/bin*.

This tool compresses the sound during conversion.

Syntax: AviToMobiclipDs <commands>

<commands> can be:

- in <inputfile.avi>
- out <outputfile.mods>
- codebook: always rebuild the audio codebook
- nocodebook : never rebuild the audio codebook
- quality: audio quality factor, default value is 128, value must be positive
- fa: use FastAudio audio codec instead of the default one. As the bitrate is fixed and no codebook is used in this codec the commands `-codebook` `-nocodebook` and `-quality` are ignored.
- debugsound <outputfile.wav>

The *debugsound* option allows you to output the resulting compressed sound in WAV format : useful when you want to have a quick preview of the quality of the compressed sound without the need of a NINTENDO DS development kit.

NOTE - for the default audio codec

A typically good trade-off between sound quality and data-rate is to use 32.768Hz sound with the default *quality* parameter (128). This should give you a sound data-rate which is about 32kbit/s.

By comparison, an average video data-rate for a full-screen (256x192) high quality (30fps) video can typically be around 400kbit/s (this depends highly on the video).

You should keep in mind the video data-rate when tuning audio quality, as audio generally doesn't use much of the available space. On the other hand, low data-rate video can make audio data-rate more significant.

If you are not satisfied with the sound quality or data-rate, you can adjust both the sampling frequency and the *quality* parameter.

The *quality* parameter ranges from 0 to 1000. Based on the value you enter the resulting data-rates range from 0.6 to 1.25 bit /sample. Low *quality* gives low data-rates; high *quality* gives high data-rates. Good trade-offs between quality and data-rate can be found around the default value (128).

If the sound quality is not good enough at 16kHz with a high *quality* parameter, we recommend increasing sampling frequency at 32kHz (but it will also increase data-rate accordingly). In the same way, if data-rate is too high, even with a low *quality* parameter, we recommend decreasing sampling frequency.

A codebook (.cbk) file is an "audio analysis" file generated the first time a video is converted. If the same file is converted another time, the codebook is not re-calculated (which saves time).

The *codebook* option forces the re-calculation of the codebook, even if the video file has not changed.

The *nocodebook* option is useful when you convert a video file which is changing many times always keeping the same audio track (typically when you are trying different video compression parameters on the same file).

If you use the *nocodebook* option, the codebook will not be recalculated every time, it will be done only the first time saving time accordingly.

NOTE - for the FastAudio audio codec

The data-rate is fixed at 1.25 bits/sample. Decompression is about two times slower than the default audio codec but the sound quality can be substantially better.

If you require better sound quality, please use this audio codec and take into consideration the slower decoding speed as well as the potentially larger resulting file size issue.

Mobiclip videos

You will find in the *Mobiclip/buid/demos/Mobiclip/data* directory the following Mobiclip resources:

- DS_up.mods
- DS_down.mods
- DS_mono.mods
- DS_stereo.mods
- DS_effectiveUsage.mods

These resources are used in the sample code described below.

Sample codes

You will find in the *Mobiclip/build/demos/Mobiclip* directory four different code samples:

- *singleScreen* is a basic code sample showing how to use the Mobiclip libraries to play a simple video file with mono sound.
- *dualScreen* is a basic code sample showing how to use the Mobiclip libraries to play two different video files (with mono sound) on the upper and lower screens.
- *singleScreenStereo* is a basic code sample showing how to use the Mobiclip libraries to play a simple video file with stereo sound.
- *effectiveUsage* is a code sample showing how to use the Mobiclip libraries to optimally play a simple video file with mono sound. It uses an alarm callback for the video display management which helps reduce frame skipping and uses the visual quality enhancements features.

These four code samples are well documented. You are invited to have a look at them to learn how to use Mobiclip.

Mobiclip libraries and prototype files

Prototype files:

- 'Mobiclip.h' in directory *Mobiclip/include/nitro*.
- 'libMobiclip.h' in directory *Mobiclip/include/nitro/Mobiclip*.

Library files:

- three versions of file 'libMobiclip.a' :
 - in directory *Mobiclip/lib/ARM9-TS/Debug* for Debug builds.
 - in directory *Mobiclip/lib/ARM9-TS/Release* for Release builds.
 - in directory *Mobiclip/lib/ARM9-TS/Rom* for Rom builds.

Using the library:

Include the file 'Mobiclip.h' in your source code and link with one the three versions of the 'libMobiclip.a' library.

For further details, see the [html documentation](#) of Mobiclip and have a look at the [sample code](#).

Annex: Video resources compression settings

Preamble:

Video sources are PAL videos (720x576 at 25fps). So resulting samples are either 25fps or 12.5fps, but note that you can go up to 30fps in single screen display, and 15fps for dualscreen.

Refer to the chapter “Mobiclip SDK For NINTENDOS DS features” in this document for more details.

DS_up.mods

Resolution: 256x192

Video bitrate: 250Kbps

Framerate: 12.5 fps

Audio: 16000Hz mono, compressed with FastAudio codec (-fa option in AviToMobiclipDs program)

Video encoding process:

- Pass 1:
 - Encoding type: Multipass first pass
 - Target Bitrate: 250
 - All other options to their default values
- Pass 2:
 - Encoding type: Multipass next pass
 - Target Bitrate: 250
- Pass 3: same as pass 2

DS_down.mods

Same settings as DS_Up.mods, but without sound.

DS_stereo.mods

Resolution: 256x192

Video bitrate: 350Kbps

Framerate: 25 fps

Audio: 16000Hz stereo, compressed with FastAudio codec (-fa option in AviToMobiclipDs program)

Video encoding process:

- Pass 1:

- Encoding type: Multipass first pass
- Target Bitrate: 350
- All other options to their default values
- Pass 2:
 - Encoding type: Multipass next pass
 - Target Bitrate: 350
- Pass 3: same as pass 2

DS_mono.mods

Resolution: 256x192

Video bitrate: 350Kbps

Framerate: 25 fps

Audio: 16000Hz mono, compressed with FastAudio codec (-fa option in AviToMobiclipDs program)

Video encoding process:

- Pass 1:
 - Encoding type: Multipass first pass
 - Target Bitrate: 350
 - All other options to their default values
- Pass 2:
 - Encoding type: Multipass next pass
 - Target Bitrate: 350
- Pass 3: same as pass 2

DS_effectiveUsage.mods

Resolution: 256x192

Video bitrate: 185Kbps

Framerate: 25 fps

Audio: 16000Hz mono, compressed with FastAudio codec (-fa option in AviToMobiclipDs program)

Video encoding process:

- Pass 1:
 - Encoding type: Multipass first pass
 - Target Bitrate: 185
 - All other options to their default values
- Pass 2:
 - Encoding type: Multipass next pass
 - Target Bitrate: 185
- Pass 3: same as pass 2