AGMVCLI v2.0

Adaptive Graphics Motion Video Command Line Interface

User's Guide

1. Introduction

The *Adaptive Graphics Motion Video Command Line Interface (AGMVCLI) is a simple command-line tool apart of the suite of software utility programs of the libagmv video codec designed to automate the encoding, decoding, and conversion process of agmv videos.*

The main caveat of AGMVCLI is that while it is a command-line tool, commands are not entered into the command-line directly. Instead, a specially formatted script file heavily inspired from the Quake 2 Cinematics video encoding tool is entered into the command-line as an argument and then used to feed data to the program.

The script file, much like Quake 2's, is very linear and cannot deviate from its set structure. All the commands inside of the file must all be put in the same line. There is only one exception where one specific section is optional, and that's specifying an audio file if a video does contain audio. The script file is composed of "tokens". These tokens serve as identifiers to common enumerations and data structures of libagmv.

The last thing worth mentioning is that, unlike other video encoders, libagmv doesn't support the 001.bmp, 002.bmp, 003.bmp naming convention. Instead, 1.bmp, 2.bmp, 3.bmp is used. In C parlance, %d.bmp, would be the format specifier required to get images into a libagmv-compliant format.

2. ***Command-Line Options***

As stated previously, an AGMVCLI script file consists of a number of “tokens”.

The table below contains the type of token at the start of the column followed by all the variants of that token type.

All AGMVCLI script files, commonly ending with a .agvs extension to signify Adaptive Graphics Video Script, must begin with the token $video, an homage to first token in Quake 2 Cinematics scripts. This will apply to all the succeeding sections as well.

Mode	Image Type	Audio Type	Optimization	Quality	Compression
ENC	BMP	WAV	OPT_I	LOW_Q	LZSS
DEC	TGA	AIFF	OPT_II	MID_Q	NONE
CONV	TIM	AIFC	OPT_III	HIGH_Q	
	PCX		OPT_ANIM	CUSTOM	
	PPM		OPT_GBA_I		
	PVR		OPT_GBA_II		
	3DF		OPT_GBA_III		
			OPT_PSX_I		
			OPT_PSX_II		
			OPT_PSX_III		

Mode: ENC -> Encode Video, DEC -> Decode Video, CONV -> Convert Video

Image Type : File extensions of various image file formats

Audio Type : File extensions of various audio file formats

Optimization: Fine tunes video encoding to specific platform or

Quality: Sets color format of video, RGB555, RGB565, or RGB665, sets leniency parameters to approximate the original image, can limit or worsen blocking artifacts, color banding, and other metrics of image quality depending on how small you want the file size to be.

Compression: Entropy and dictionary based coding to apply to bitstream’s of libagmv audio and video data.

3. ***Simple Video Encoding***

Here is the basic structural layout of an AGMVCLI script file to encode a simple, soundless video.

$video (MODE) (FILENAME) (DIRECTORY) (BASENAME) (IMAGE TYPE) (START_FRAME) (END_FRAME) (WIDTH) (HEIGHT) (FPS) (OPTIMIZATION) (QUALITY) (COMPRESSION)

$video ENC simple.agmv simple agmv_frame_ BMP 1 200 320 240 24 OPT_I MID_Q LZSS.

The filename refers to the name that you want the output file to be.

The directory is the folder where your images are. The token can be set to “cur” if all of the images are located in the local directory of the AGMVCLI executable.

$video ENC simple.agmv cur agmv_frame_ …….

The base name is the prefix that all the images contain, such as frame1.bmp, frame2.bmp, frame3.bmp. The base name would be frame in this instance.

The start and end frame can be set to start or end anywhere in the image sequence. You can start at 5 and end at 10 even if it goes from 1 to 100.

FPS refers to the frame rate of the video, frames per second to be exact.

4. ***Simple Video Decoding***

Here is the basic structural layout of an AGMVCLI script file to decode a video.

$video (MODE) (DIRECTORY) (FILENAME) (IMAGE_TYPE) (AUDIO_TYPE)

$video DEC cur simple.agmv BMP WAV

Hopefully, this section is fairly straightforward and doesn't require much explanation. To decode an agmv video using AGMVCLI, the decoder mode, set by DEC, must be set. If the video doesn't contain audio, the audio type isn't required, and the following structure will work just fine.

$video DEC cur simple.agmv BMP

5. ***Adding Audio Tracks to Video***

The structure of an AGMVCLI script file containing audio doesn't deviate from a script file that doesn't contain audio. The only addition is the optional "-v" token followed by your audio file.

Here is the basic structure of an AGMVCLI script file that contains audio.

$video (MODE) (FILENAME) (DIRECTORY) (BASENAME) (IMAGE TYPE) (START_FRAME) (END_FRAME) (WIDTH) (HEIGHT) (FPS) (OPTIMIZATION) (QUALITY) (COMPRESSION) -v (AUDIO_FILE)

$video ENC simple.agmv simple agmv_frame_ BMP 1 200 320 240 24 OPT_I MID_Q LZSS -v simple.wav

A legacy feature of AGMVCLI is the ability to load raw signed 8-bit PCM audio files by specifying the filename along with the sample rate of the audio. The example case below has the sample rate set to 16,000. For the GBA, the audio should be between 8,000 and 32,000 for AGMVCLI to accept it.

$video ENC simple.agmv simple agmv_frame_ BMP 1 200 320 240 24 OPT_I MID_Q LZSS -v simple.raw 16000

6. ***Advanced Quality Settings***

libagmv comes preset with three quality settings, Low, Medium, and High, that determine the internal color formatting of the video and sets the leniency bar for the various block size copying and motion vectors that can either be highly accurate or approximate these blocks from previous frames. libagmv also has a structure that can manually set all of these values. To set these values in AGMVCLI, the quality mode must be set to CUSTOM, and a file named agmvcli.quality must be present in the same directory of the AGMVCLI executable so that It can parse the file for these manual values.

There are currently eight different modes a block can take and four block sizes in libagmv v2.0. The following quality file reflects this. Each row is one of the block sizes, and each column is setting the bar for each possible mode that the block can take.

The first row is a 4x4 block containing 16 pixels, the second is an 8x4 block containing 32 pixels, the third is a 2x4 block containing 8 pixels, and the last is an 8x8 block containing 64 pixels. Go ahead and play around with the values. The closer the value is to the total number of pixels, the more accurate but larger the output file will be. The further, the greater the approximation, smaller file size, but the less accurate the video will become. Don't change the final two columns of each row unless you study the different block modes, "block flags" in libagmv. It's too complicated to explain, but values 2 and 4 in their respective locations work just fine.

```
14 14 14 14 14 14 2 4
28 30 28 28 26 26 2 4
8 7 7 8 7 8 2 4
58 56 56 56 54 54 2 4
```

8. ***Converting AVI videos to AGMV***

It's important to note before demonstrating the structural layout of the conversion process that AGMVCLI, to date, only supports AVI videos that use the DIB RGB24 video codec, that is to say they are essentially raw 24-bit per pixel BMP images, and raw wav PCM audio. Here is the command to convert an MP4 video to this format using a tool like ffmpeg,

ffmpeg -i input.mp4 -c:v rawvideo -pix_fmt rgb24 -c:a pcm_s16le output.avi.

Here is the basic structure of an AGMVCLI script file that converts an AVI video to an AGMV video.

$video (MODE) (AVI)(AGMV)(FILENAME) (OPTIMIZATION) (QUALITY) (COMPRESSION) (OPTIONAL DITHER)

$video CONV AVI AGMV example.avi OPT_I HIGH_Q LZSS

The structure bears many similarities with the typical encoding procedure discussed earlier on, but the encoder scans the AVI to extract all the important information that the encoder needs so that you aren't burdened with providing it, but it removes the flexibility to determine the start and end frame to control what section of the video gets encoded.

Applying a dither to the corresponding image data requires the optional token **D**

$video CONV AVI AGMV example.avi OPT_I HIGH_Q LZSS D

Similar to libsmacker, applying a dither onto the image data, while greatly increasing the total file size, can reduce some of the nastier visual artifacts caused by the encoder. That's why it's an optional token. Certain videos look fine without the dither, but others benefit visually from having it. It's up to you to decide whether your video needs it.

9. ***Converting AGMV videos to AVI***

Here is the basic structure of an AGMVCLI script file that converts an AVI video to an AGMV video.

$video (MODE) (AGMV)(AVI)(FILENAME)

$video CONV AGMV AVI example.agmv

Converting an AGMV video to an AVI is much simpler than the inverse process. Just remember that it's a to-from structure, so unlike last time, you are specifying to the encoder that you are going from AGMV to AVI and not AVI to AGMV.