Comparison of Containers
AVI, OGM, MKV

Last Modification: April 17, 2006

Contents

1 Scope

This scope of this document is to introduce the AVI, OGM and MKV containers to you, comparing their capabilities, and their advantages and disadvantages over each other. I won't talk about MP4. The reason is simply that I don't have the knowledge necessary to make a comparision between MP4 and other containers which anyone would take serious. Not mentioning MP4 does NOT mean that I don't like it, that it is not a competition for others, nor does it mean anything else you might come up with, except for that I don't know much about it.

As most users on forums trying to propagate one certain container are either the developer of the corresponding container or are trolls, you should read everything twice before believing it. Most users trying to participate in such dicussions did not code parsers or muxers for any of those containers, especially not from scratch, and yet talk about the complexity of making one. As you all know, virtually every piece of software is more or less buggy, meaning that one user reporting problems with playback or just muxing means exactly nothing. You'll find such reports for each software, and for each container, or for each other appilcation on this world.

Also, a lot of people having used something for years will never admit that whatever it was is or was bad (this is simply psychology).

As I am fed up with reading disinformation on forums, I decided to write this document.

2 Basic Overview

Property	AVI	MKV	OGM
Maximum file size	infinite*	infinite	infinite
Multiple Audio streams	yes	yes	yes
VBR audio	yes	yes	yes
VFR streams (e.g. Vorbis audio)	yes**	yes	yes
Unicode Subtitles	yes	yes	no
Multiple Subtitle streams	yes	yes	yes
Chapters	no	yes	yes
Overhead	medium	low	large
Available documentation	ok	ok	not existing
Still worked on	yes	yes	no
copy /b - Joining	no	yes***	sometimes
compatibility to hardware players	some	none	none

Note:

* an infinite file size does not really mean unlimited, but rather means that the maximum file size is a great multiple of the size of the largest existing hard discs (e.g. some billion gigabytes in case of AVI).

** *ffmpeg* is storing Vorbis in AVI files in a way that allows proper playback, but causes an extreme amount of overhead, far more than OGM

*** Although you can join MKV files using copy /b, not all tools supporting Matroska files will correctly read such files. Playback on Windows works with Haali Media Splitter.

As you can see, OGM offers only a few advantages over AVI and none over Matroska, and that AVI also has a few advantages over OGM.

3 A Deeper Look Into it

3.1 Maximum file size

The maximum size of **AVI** files is defined by the index. Signed 64 bit values are used as basic offsets to which 31 bit signed values are added. The most significant bit of the corresponding 32 bit value is used as deltaframe flag.

That means the last position the index can point to is $2^{63} - 1 + 2^{31} - 1$

As to MKV files, the current limit is the segment size, which can be a 64 bit EBML integer, meaning that its maximum value is $2^{56} - 2$. However, in the unlikely case that larger files are needed, the maximum length for EBML integers can be easily increased without breaking backward compatibility. 64 bits have merely been chosen because current CPUs can easily handle 64 bit values, but not larger values without some additional effort.

As OGM files don't use anything compareable to an index or a top level structure, their size can be unlimited.

3.2 Multiple Audio Streams

Don't use Windows MediaPlayer. Also, if you are using AVI files, you have to use Open-DML (and not AVI 1.0) if there are more than 10 streams in the file. Otherwise, the flawed Microsoft AVI splitter won't handle them.

Concerning MKV, don't use Windows Mediaplayer either.

The OGM splitter tries to support Windows MediaPlayer by including a stream selector, sometimes resulting in problems if the audio streams do not use the same format. This is actually funny because Haali Media Splitter shows that this concept can work out very well.

3.3 VBR audio

A very common myth is that AVI can't handle variable bitrate audio streams, and that MP3-VBR-in-AVI was a hack. Don't believe everything people claiming other containers to be more modern are trying to tell you. Virtually every video stream you'll find in AVI files has variable bitrate, and seeking works fine. The same method for seeking (= frame-wise seeking) can be used on variable bitrate audio streams as long as the duration of one frame is constant, as it is the case for MP3 or AAC (or HE-AAC). The method used for seeking does not depend on the stream type.

However, tools relying on ACM might not seek properly in VBR streams. The reason is that the AVIFile interfaced in use uses an (in my opinion) awkward interpretation of the AVI headers and the AVI index: It assumes that even a "complete" (independant) chunk which is larger than 0 bytes can contain no data, so that seeking is not working. Thus, while playing AVI files containing VBR audio is no problem with any player using DirectShow or simulating its behaviour with such files, editing such files might be tricky.

OGM and Matroska won't have such problems, once there will be editors handling those containers (currently VirtualDubMod can handle OGM and MKV files to some extent, but its support for those containers is not complete).

3.4 VFR streams

When storing such streams in AVI files, you need to define artifical temporal frames of constant duration. ffmpeg is using an artificial duration of 1/750 seconds when storing Vorbis audio in AVI, meaning that an audio frame of 21 milliseconds will be followed by 15 silent frames, causing between 8 and 32 bytes of overhead each, depending on the target AVI type).

Matroska can easily work with VFR audio and Video without causing such amounts of additional overhead, whereas OGM works with VFR audio, but not with VFR video. The container is theoretically capable of it, but no muxer can create such files. Like Matroska, OGM does not have more overhead for such streams, compared to CFR streams.

3.5 Subtitles

Selectable subtitles work in all 3 containers. However, OGM cannot do Unicode subtitles, meaning that combining french and russian will be hard. Even just rendering french subtitles on a russian Windows installation or vice-versa can be problematic. Also, OGM only works with SRT subtitles, whereas AVI additionally can be used with SSA and Matroska with SSA and VobSub subtitles.

As AVI files are limited to 100 streams, you might claim that this is a real limit, but it will suffice usually. Especially, DVDs can only contain 8 audio tracks and 32 subtitle tracks, so this limit is non-existant when making backups of DVD-Video discs.

However, Standalone MPEG4 players that can read AVI files will usually not be able to handle selectable subtitles in AVI files. Standalone players reading

OGM or MKV are very rare.

3.6 Overhead

The following test files have been used:

Movie 1 (A Babylon 5 - Episode):
41m 57.080s, 2xAC3 @ 384 kbps

Movie 2 (Lord of the Rings - The Two Towers):
3h 45m 47s, 2x AC3 @ 384 kbps, 1x DTS @ 754,5 kbps

Movie 3:
3h 45m 47s, 1x MP3 VBR (128 kbps)

Movie 4:
3h 45m 47s, 1x Vorbis

Movie 5:
1h 30m 28s, 3x HE-AAC (2ch, about 70 kBit/s each)

Muxing applications:

AVI	AVI-Mux GUI 1.16.8	5 AC3 frames per Chunk, 20 DTS frames per Chunk
MKV	AVI-Mux GUI 1.16.6	300ms per lace, Cues for video and audio, no Seek entries for Clusters
OGM	VirtualDubMod 1.5.10.1	

Overhead test results:

	AVI		OGM	MKV
	Open-DML	AVI 1.0		
Movie 1	1,749,393	2,587,493	8,570,928	1,139,324
Movie 2	10,598,516	n/a	52,604,141	7,054,227
Movie 3	14,948,112	22,385,528	24,148,369	6,081,726
Movie 5	8,188,807	12,145,331	n/a	3,005,166

As Vorbis audio in AVI is not yet really usable, and as laced Vorbis streams in Matroska files seem to be problematic sometimes, I'll indicate overhead values for Matroska for different lace settings, as well as for OGM:

	MKV			OGM
	no lacing	50 ms per lace	100 ms per lace	
Movie 4	12,550,903	7,477,934	6,870,450	21,751,467

As you can see, even the version without any lacing contains less overhead than the OGM file.

Now you might wonder why the overhead of OGM files is that high. To explain this, it is necessary to look closer at the (very simple) main structure of OGG and OGM file formats.

Each OGG file consists of pages. Each page header contains 28 bytes, plus a list of segment sizes. One segment can have a size of up to 255 bytes, one page can contain 255 segments. In other words, for each 255 bytes of video or audio data, you get at least one byte of overhead, as well as 28 bytes for each 4 kByte.

With a default page size of 4 kByte, you get about 1,1% - 1,2% of overhead. The overhead could be decreased by up 50% if larger pages would be used. As demonstrated later, at least pages containing video could be larger without hurting the file.

3.7 Binary Joining

Joining 2 AVI files using copy /b will result in an invalid file

Joining 2 OGM files using copy /b might work or not: If the video/audio formats are compatible, the output file could work.

Joining 2 Matroska files using copy /b will result in a file containing 2 segments, which is valid, but only supported by a few tools.

4 Developers' Points of View

4.1 Documentation

While there is a sufficient amount of documentation available on AVI and Matroska, there is virtually none for OGM, except for the source code. Writing an OGM parser or an OGM muxer from scratch, as I did for AVI, Matroska and OGG/Vorbis, is a problem, as a verification of OGM files is not possible. Something like a 'valid OGM file' is just not defined.

4.2 Index

People propagating OGM usually claim that OGM not using an index is an advantage, because an index that can't be lost can't render a file useless by being lost. I'll explain why this is not an advantage at all.

Indexes are usually lists containing timestamps (explicite or implicite ones) and positions within a file to make seeking easier. The question to be looked into a little deeper is what is happening if the index of a file gets damaged.

In AVI files, the index is essential. It does not only contain a list of all frames of all streams, but also contains keyframe information. With an index being lost, keyframe information is lost along with it. Even though it is still possible, seeking may get slow, and reconstructing the keyframe flags can be easy or not at all easy, depending on the stream format. For example, if the frame type is coded in the raw frame data, like in MPEG4, reconstructing the index is not too hard to implement.

The index of Matroska files ('Cues') does not contain information necessary for replay. Cues allow easier seeking, but losing the Cues - Element of a Matroska file does neither cause the loss of keyframe information nor the loss of the ability to properly replay or easily reconstruct a file. That means: If the index is there, easy seeking is possible. If the index is lost, the 'OGM' - way of seeking is still available.

That's why claiming OGM not using any kind of index is an advantage is simply wrong. On the contrary, when trying to seek in an OGG or OGM file, it is required to do a binary search on OGG pages, where the beginning of each page has to be found by using trial & error during that binary search. This is equal to seeking in a Matroska file of which the index is lost. Also, seeking will work faster if the correct target can be found with one single seek operation on the file (the index can be kept in memory) when users use media that is slow in seeking, like low quality DVD-R.

4.3 OGM page sizes of only 4 kByte are not useful

The idea of small pages like this is that a page header being lost (error in transmission, destroyed data on a CD) should only destroy a piece as small as possible of a video or audio file.

While this idea makes sense for audio files at low bitrates, it is stupid for videos. If a frame is lost, all frames till the next keyframe are screwed up anyway, meaning that at a data rate of 500 kBit/s (pretty low) and a keyframe rate of 1/25 (pretty low), up to 80 kB of video data are useless, whether or not it is entirely lost. Thus, using the maximum page size, and starting a new page with each keyframe, would significantly reduce overhead, without descreasing error resilence.

Unfortunately, the OGG page size cannot be simply increased: It is hard-coded into libogg. Thus, changing the page size in libogg would break low-bitrate Vorbis audio muxing. It would have to be made configurable, which no one seems to be interested in. At the moment, only AVI-Mux GUI is capable of creating OGG files (namely OGG/Vorbis files, when demuxing Vorbis streams from Matroska files) with configurable page size, as it is using an indepentant implementation for OGG file reading and writing.

4.4 Dependencies of Container and Stream formats

One thing making the life of developers easier is a container with which handling files does not require any knowledge about stream formats.

4.4.1 AVI

Reading AVI files is theoretically possible without any knowledge about the format of streams, however, real life is different. The large number of flawed muxing programs writing weird streams and/or weird headers requires that certain stream types, especially MP3 and AC3, get special treatment, to ensure proper playability of output files. Also, the way to store subtitles in AVI files, inspired by 'Gabest', does require a complete SRT/SSA parser to handle SRT/SSA - subtitles in AVI files.

4.4.2 Matroska

Whereas muxing different sources to Matroska files requires knowledge about the formats to mux (e.g. to find the correct CodecID), transmuxing from Matroska to Matroska does not require knowledge about the contents, unless suboptimal or invalid files are to be supported to a great extent (for example to resolve laces of AC3, MP3 or DTS audio streams with DefaultDuration being set to 0, or, even worse, when rebuilding the original timestamps of frames of a laced Vorbis stream is necessary). Joining Matroska files only requires knowledge about streams in a few situations, e.g. when joining Matroska files with SSA subtitles with incompatible style defintions, which can be done by rewriting the style and subtitle definitions.

4.4.3 OGM

The situation with OGM is worse than with AVI or Matroska. One critical issue is the GranulePos of OGG Page headers, which is similar to a timestamp. Unfortunately, the specification is rather unspecific here: Although only the heir of the realm of idiots would use a logarithmic scale for timestamps, it would not violate any point of the OGG page header definition (OGG/Vorbis is not affected by this, as the way to store Vorbis in the OGG container is clearly defined and does not leave space for such funny ways to read the specification). Basicly, some knowledge about each Codec to be supported is necessary, even for remuxing OGM files.

4.5 Using those containers in own applications

4.5.1 AVI

In short: The Microsoft libraries for AVI file access are broken, in reading as well as in writing AVI files. Reading and Writing proper AVI files requires that other code is used, or that such code is build from scratch. The documentation available on AVI files is sufficient for that. Just a few examples what is broken:

- Reading of AVI 1.0 files with more than 10 streams is broken
- Reading of Open-DML files with more than 1 audio stream is sometimes broken (and always broken with at least 4 audio streams) when the standard index blocks are too large
- Reading of Open-DML files with MP3 audio fails when the suggested buffer size for that stream is 0 (this is valid and is supposed to indicate that the reader/player has to determine a reasonable buffer size on its own)
- Playing an AVI file where the first chunk is a video chunk doesn't work when the audio format is PCM

4.5.2 OGM

No documentation, no one to ask, just the source code of the filters. That is a rather poor base for code using that container, even though a few applications are able to use OGM (like mkvmerge and VirtualDubMod).

4.5.3 Matroska

The official 'libmatroska' can be used, however, it is not very well documented. As the author can be reached via email and on IRC, asking himself is not a problem in the case that the usage of the code is not clear.

The specification is sufficient to write own parsers und muxers from scratch, as several people have already proven by doing that. However, a Matroska parser and/or muxer supporting everything Matroska supports will be quite complex, compared to an AVI reader/muxer.

4.6 Using Mode 2 - Form 2 - CDs

"Mode 2 - Form 2" is a way to store data on a CD, removing the 3rd layer of error correction (similar to Audio CDs, even though Audio CDs are NOT using Mode 2 Form 2), and storing 800 MB on a normal 80min CD-R/W. Less error correction of course means less protection, leading to people claiming that containers like OGM, AVI and MKV need an own layer of error detection. I'll now explain why this claim is merely based on a bad feeling in the stomach rather than on facts. Note that I am speaking about detection, not about correction. If you want more error correction, then don't use Mode 2 - Form 2, but use DVDs instead (whose error correction capabilities are far superior to those of Mode 1 - CD).

One `Sector` of a CD in Mode 2 Form 2 - Format contains 2324 bytes of data, some header information and a 32 bit CRC. The CRC is not mandatory, but is usually stored, as the space is reserved and cannot be used for any other purpose. That means, it is easily possible to determine whether a piece of 2324 bytes of data starting at a sector boundary is broken or not.

Now there are 2 possibilities: First, the container could use CRCs spanning over larger pieces of data. Then, those CRCs are not useful, because the CRCs included in Mode 2 - Form 2 tracks allow more precise conclusions as to where data has been damaged. So the container CRCs would not give any additional information.

Second, the container could use CRCs spanning over smaller pieces of data, much smaller than 2324 bytes. To give any new information about broken data, the maximum size would be half of that. In that case, an overhead of 4/1162 = about 0,25% would be added. Considering that some containers allow muxing files with an overall overhead below this value, the overhead would be more than doubled just to allow locating broken data more precisely.

OGM, with a CRC over pages of 4 kBytes, obviously belongs to category (1), whereas CRCs in Matroska are not mandatory, usually occur only over much larger pieces of data (some hundred kBytes), and thus do only waste about 1 byte per 10 - 100 kBytes of data.

5 Summary

As you have seen, those properties of OGM used by people who propagate it are no real advandage, and at least one major drawback could be easily fixed to some extent without breaking compatibility to anything. But no one seems to care to fix it. It seems that people who could fix don't care enough about OGM to really do it.

Finally, here the excerpt of a 'discussion' between one of the most famous OGM Zealot and other people:

```
Zealot: "Who would put ac3 or dts into ogm btw.? OGM is meant to be
a container for ogg vorbis sondtracks and video as well as subtitles."
Someone else: "So you acknowledge the fact that OGM is just a technology
demonstration to advertise Vorbis ? Not a general purpose container ?"
Zealot: "You know very well that OGM is a general purpose container..."
```

As you can see, not even OGM Zealots trying to defend OGM at all costs (again, OGM, I don't speak about OGG here!) can write reasonable sentences not contradicting each other.

While AVI does not support everything you might expect from a modern container, it is still the only one widely supported, especially on MPEG4 standlone players, so it won't vanish too soon. It can still be enhanced.

As to Matroska, the only bad thing is that people ask me to add OGM read support to AVI-Mux GUI, to be able to easily remux their OGM files to Matroska.

6 Questions, Comments, Contact

If you have any questions concerning this document, if you have comments, additions, if you have found an error, or if you want to contact me for whatever reason, send a mail (include 'containers' in the topic!) to

`alex@alexander-noe.com`

in german, english or french. Please don't use a machine translator.