

Media Formats

- Sound
 - WAV (uncompressed, PCM)
 - MIDI
- Image
 - BMP
 - XML dia
- Video
 - MPEG

Sources

- WAV Sound
 - Chapter 4, main text
 - Kientzle, T. (1998). A Programmer's Guide to Sound. Addison-Wesley.

WAV file format

- A file header
- Binary samples
- Variable file format
 - Compression is possible
 - Uncompressed format also possible (“PCM”)
- In the following, we will discuss one type of WAV file format
 - If you create a file in this format, it will be interpreted as a WAV file
 - There are other formats (e.g., different headers, compressed data) that are also interpreted as WAV files

WAV file header (PCM)

```
/* My assumptions about data type lengths */  
#define      int8      char  
#define      int16     short  
#define      uint16    unsigned short  
#define      int32     int  
  
/* WAV file header;  
Mostly from Kientzle (1998). */
```

```
typedef struct {
    int8 ChunkType1[4]; /* 'R', 'I', 'F', 'F' */
    int32 length; /* int bytes, length of everything after this in file */
    int8 ContainerType[4]; /* 'W', 'A', 'V', 'E' */
    int8 ChunkType2[4]; /* 'f', 'm', 't', ' ' */
    int32 FormatChunkDataLength; /* 16 */
    FormatChunkData fcd; /* 16 bytes long */
    int8 ChunkType3[4]; /* 'd', 'a', 't', 'a' */
    int32 SoundLength; /* in bytes */
    /* After this, SoundLength bytes of data in file */
} WavHeader;
```

```
typedef struct {  
    int16 CompressionCode; /* 1 for PCM */  
    int16 NumberOfChannels; /* 1 for mono */  
    int32 SamplesPerSecond;  
    int32 AvgNumberBytesPerSecond;  
    int16 BlockAlignment;  
    int16 SignificantBitsPerSample; /* 8 or 16 */  
    /* 16 bytes total */  
} FormatChunkData;
```

dog.wav from Chapter 3

- <http://www.cprince.com/courses/cs3121fall02/lectures/media/dog.wav>
- File size: 29764 bytes

FormatChunkDataLength= 16

CompressionCode= 1

NumberOfChannels= 1

SamplesPerSecond= 22050

AvgNumberBytesPerSecond= 22050

BlockAlignment= 1

SignificantBitsPerSample= 8

sizeof header= 44

sizeof fcd= 16

SoundLength= 29720

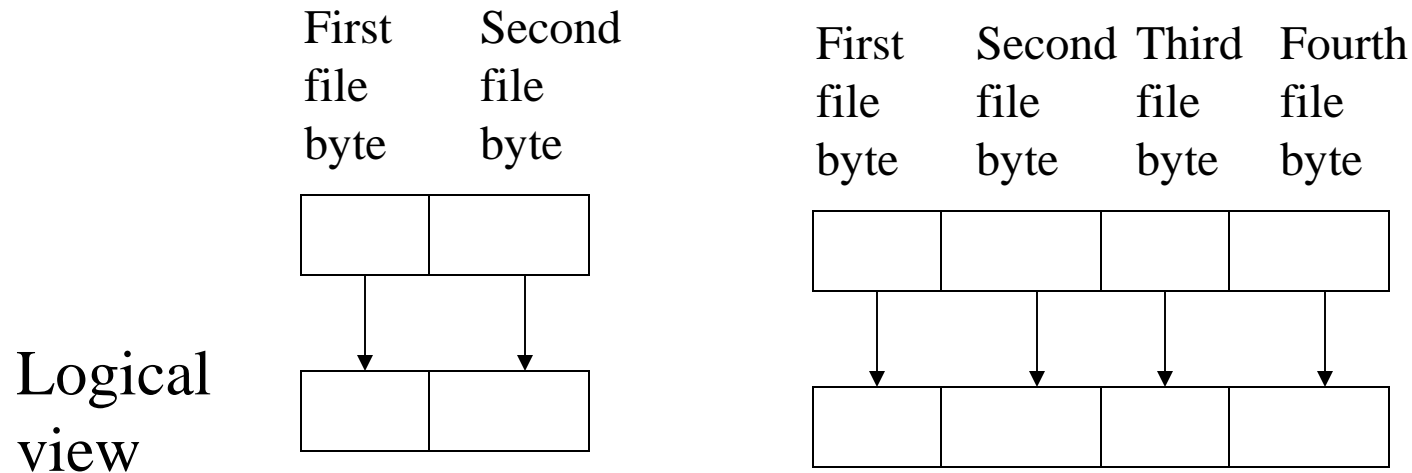
length= 29756

Text vs. Binary Files

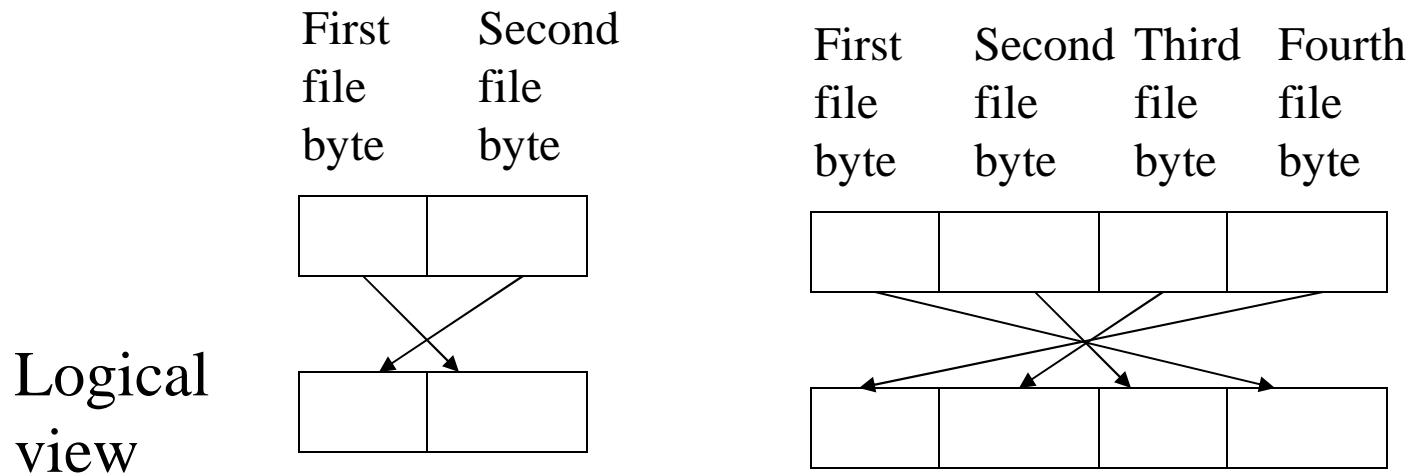
- A text file is usually one that can be displayed in a word processor, or text editor
 - The file [example.txt](#) is a text file
 - (No particular reason the .txt is needed!)
 - .cpp, .c, .h program code source files are text files
- Binary files
 - Are usually not viewable, interpretable in a text editor
 - On UNIX systems, “cat -v filename” will display in a ‘printable’ format
 - These files need special interpretation
 - E.g., integers stored in binary format

Binary Integers

- MSB: Most significant byte first



- LSB: Least significant byte first



WAV files

- WAV files are ‘binary’
 - Integer values such as lengths are stored in binary integer format
- Header integer values
 - Stored in LSB format
- Data samples
 - E.g., 8 bit or 16 bit integers
 - Stored in LSB format
 - Not relevant for 8 bit samples
 - *Unsigned* integers
 - E.g., values between 0 ... 255
 - What does this mean for amplitude values?

MIDI: Sources

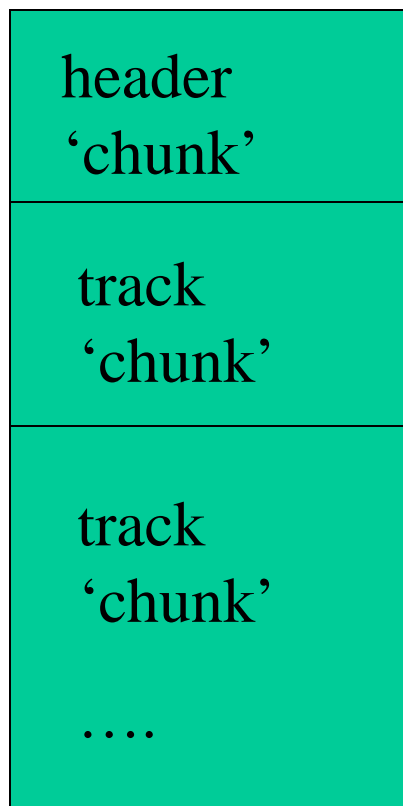
- Kientzle, T. (1998).
- http://crystal.apana.org.au/~ghansper/midi_introduction/

MIDI sound

- **MIDI: Musical Instrument Digital Interface**
 - Used to specify music
- **Abstract representation**
 - Part of the standard is sample based
 - Part based in specification of timing and instrument types
- **General MIDI**
 - Specifies 175 standard instrument sounds
- **MIDI files are organized as a series of tracks**
 - Each track might store piano, trumpet, and flute parts of a piece of music as separate tracks
 - Tracks contain events
 - Events specify a musical action; e.g., key press, or release
- **Three types of files**
 - 0) one track only
 - 1) multiple tracks, to be played back simultaneously
 - 2) multiple tracks, but no timing relation assumed

MIDI files

- Binary file

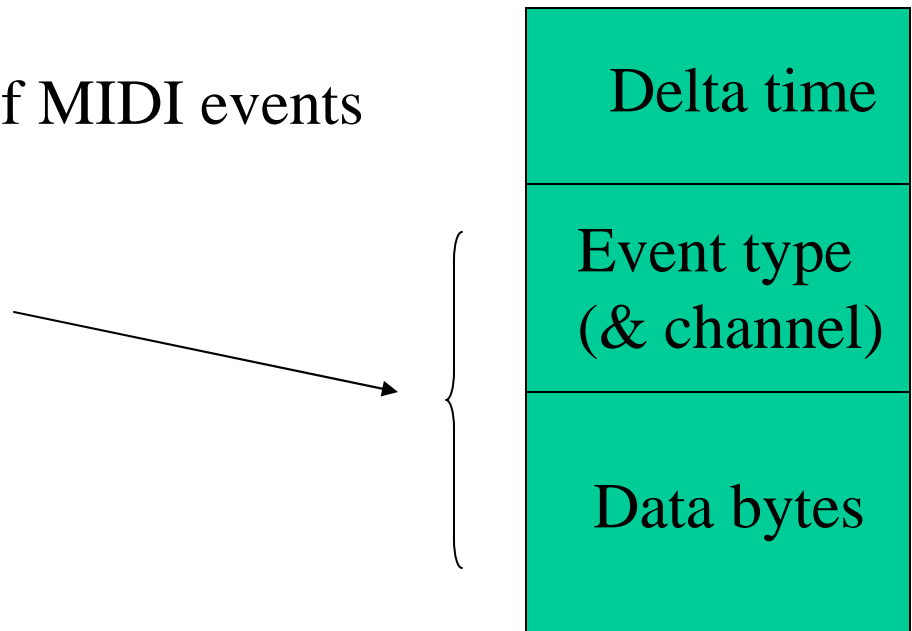


MIDI file format

- Header
 - Four byte ‘MThd’ chunk name
 - Four byte MSB integer
 - 6 bytes of format specification
 - file type (2 byte, MSB) (0, 1, 2)
 - number of tracks (2 byte, MSB)
 - time format (2 byte, MSB)

- Track
 - List of MIDI events

- Event



Variable
length integer

Aka. *Status*
byte (repeated
status bytes
can be
omitted)

Delta time (unit: ticks)

- Amount of time that separates this event from the previous event
- *variable length* integer
 - Positive values only
 - 1 to 4 bytes
 - Last byte of integer stored with a 1 bit in high-order bit, others stored with a 0 in high order bit
- Interpretation of this time in terms of a duration depends on time format in header of file
 - -ve time format value: SMPTE (Society of Motion Picture and Television Engineers) standard
 - Tick values specified in hh:mm:ss;ff format
 - Time format specified as frames/second & ticks/frame
 - +ve time format value: musical tempo
 - Ticks per beat given in format value

Specification of Instrument Sounds

- **General MIDI**
 - Channels 1-9, 11-16
 - melody channels
 - One of 125 instruments (Table 22.5, Kientzle, 1998)
 - Channel 10
 - Rhythm channel
 - One of 47 instrument sounds (Table 22.6, Kientzle, 1998)
- **DLS: Downloadable Samples Standard**
 - Method for storing MIDI instrument sounds in a file
 - Format based on WAV

Image formats

- XML dia format (abstract)
 - Supports circuits, networks, flowcharts
 - Enables some new types of drawing objects to be defined
 - dia program generates gzipped xml
- BMP format (sample-based)
 - Device-independent “bitmap” files
 - Bitmap = sample-based
 - Often used in Microsoft operating systems
 - Based on RGB color model

XML dia Format

- Sources
 - <http://www.gnome.org/gnome-office/dia.shtml>
 - Dia home page
 - <http://www.lysator.liu.se/~alla/dia/>
 - Book chapter on dia
 - <http://www.togaware.com/linuxbook/dia.html>
- Example
 - dia-box.xml
 - <http://www.cprince.com/courses/cs3121fall02/lectures/media/dia-box.xml>
- Shape extension example
 - <http://www.lysator.liu.se/~alla/dia/custom-shapes>
 - dia-shape-ns.xml
 - <http://www.cprince.com/courses/cs3121fall02/lectures/media/dia-shape-ns.xml>

BMP Format

- Sources
 - Steinmetz, R. & Nahrstedt, K. (2002). Media Coding and Content Processing. Volume 1. Prentice Hall.
 - <http://www.dcs.ed.ac.uk/home/mxr/gfx/2d/BMP.txt>
 - <http://www.fortunecity.com/skyscraper/windows/364/bmpffrmt.html>

BMP

- Header followed by a data region



BMP

- Header
 - Sizes
 - Of file, in bytes
 - Width & height of image– in pixels
 - Color
 - Bits per pixel (color depth): 1, 4, 8, 24
 - Color lookup table (CLUT)
 - Compression method (omit for now)
- Data region
 - Value of pixels in a line
 - Lines of image stored in reverse order

Color Lookup Tables (CLUT) - 1

- Provides a way to specify color with a smaller number of bits
 - Each entry in the CLUT will be a full color specification for the present system hardware
 - E.g., 24 bit RGB
- 1 bit: needs 2 entries in CLUT
 - Black, white

0 in bitmap

white (24 bits)

1 in bitmap

black (24 bits)

Color Lookup Tables (CLUT) - 2

- 4 bits: Needs 16 entries in CLUT
 - Each of the possible 4 bit values indexes into the CLUT
 - Each entry in the CLUT is full RGB representation (e.g., 24 bits)
- 8 bits: Similar, but 256 entries in CLUT
- 24 bits: No clut, just RGB specification

CLUT Example

- How much information in a BMP file is needed to represent a 10x15 pixel b&w image with 1 bit pixels. Include only the pixel data and the CLUT in your calculations (assume 24 bit RGB in CLUT entries).
- Bytes for pixels of image
 - 10x15 image = 150 pixels
 - 150 pixels * 1 bits/pixel = 150 bits
 - 150 bits/(8 bits/byte) = 18.75 bytes (need to round up)
 - $\lceil 18.75 \rceil = 19$
- Bytes for CLUT
 - 2 entries in CLUT (one for black, one for white) * 3 bytes for RGB per entry = 6 bytes
- Total
 - 19 + 6 = 25 bytes

Example without the CLUT

- How much information in a BMP file is needed to represent a 10x15 pixel b&w image, assume 24 bit RGB needed to specify colors of pixels (no CLUT)
- Bytes for pixels of image
 - 10x15 image = 150 pixels
 - 150 pixels * 3 bytes/pixel = 450 bytes
- Comparison
 - 26 bytes with CLUT
 - 450 bytes without
 - $450/26 = 17.3$
 - Savings of space:** factor of 17

Media File Formats

- Image formats:
 - XML/dia, BMP; Others include: JPEG, GIF, PNG, TIFF
- Sound formats:
 - WAV, MIDI; Others include MP3, AIFF, AU
- Video formats:
 - AVI; Others include MPEG, Quicktime, Realmedia

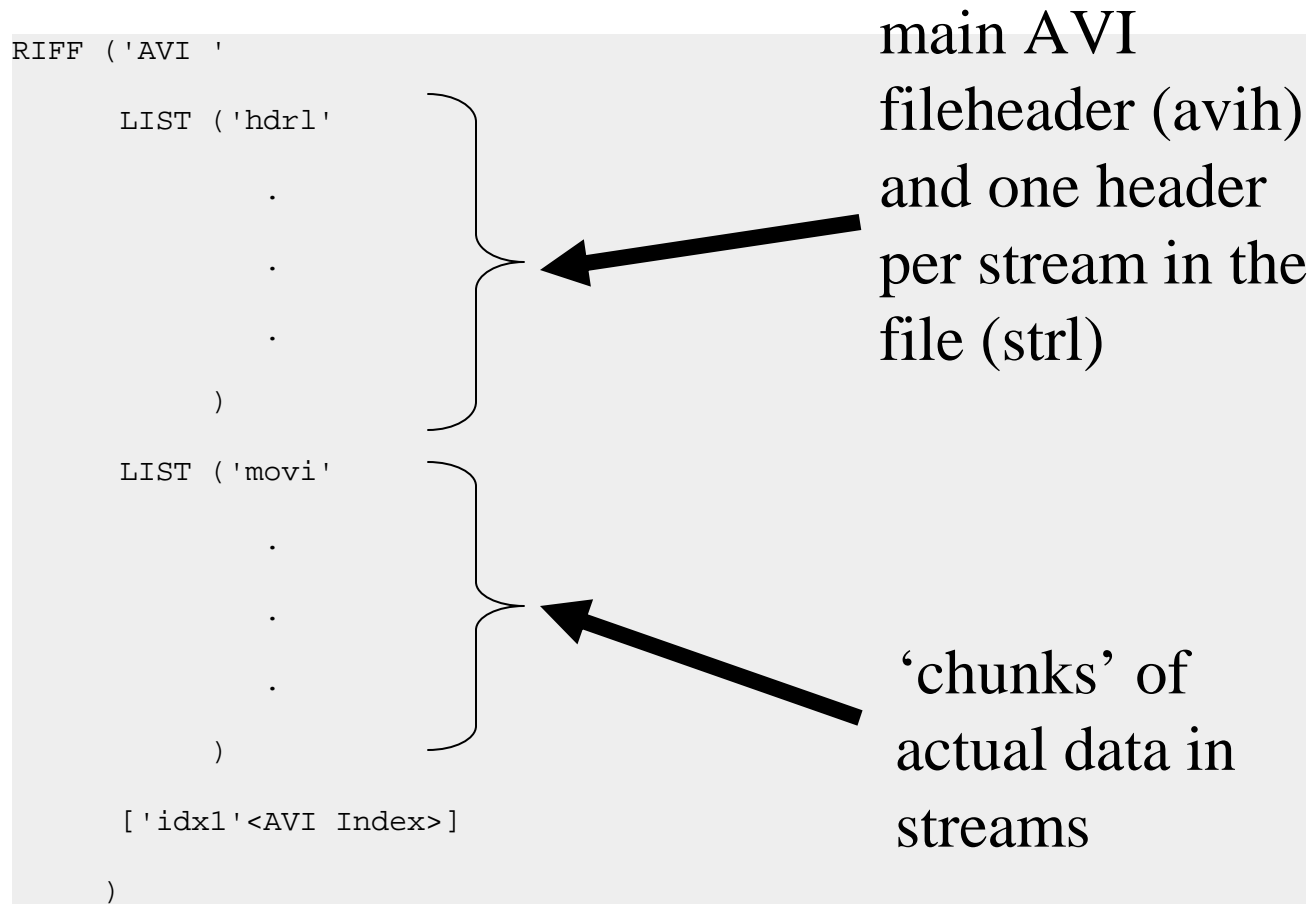
Video Formats

- Sources
 - Chapter 10 of lab text, pp. 213-223
 - Chapter 4 of main text, pp. 209-212
 - Steinmetz & Nahrstedt (2002), Chapter 7
 - AVI format reference:
 - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dx8_c/directx_cpp/html/aviriffilereference.asp
- Digital video formats
 - More complex than formats discussed so far
 - Typically include two data streams: video & audio
 - More data (in video); compression is more important
 - Quality of data required in applications may vary widely
 - E.g., Commercial quality vs. home quality

AVI files

- AVI: Audio-video interleave
- RIFF file format (same class as WAV)
- Used for video applications (capture, edit, playback)
- Can be compressed (“raw” video) or uncompressed
- Can contain ≥ 1 stream of data
 - Video, audio, text

AVI File Structure



Main header

```
typedef struct {  
  
    DWORD dwMicroSecPerFrame;  
  
    DWORD dwMaxBytesPerSec;  
  
    DWORD dwReserved1;  
  
    DWORD dwFlags;  
  
    DWORD dwTotalFrames;  
  
    DWORD dwInitialFrames;  
  
    DWORD dwStreams;  
  
    DWORD dwSuggestedBufferSize;  
  
    DWORD dwWidth;  
  
    DWORD dwHeight;  
  
    DWORD dwReserved[4];  
  
} MainAVIHeader;
```

- **dwMicroSecPerFrame**
 - Number of microseconds between frames.
- **dwMaxBytesPerSec**
 - Maximum data rate of the file: number of bytes per second.
- **dwFlags**
 - Contains any flags for the file; e.g., does the file contain an index chunk, or is the file *interleaved*, copyright
- **dwTotalFrames**
 - number of frames of data in the file.
- **dwInitialFrames**
 - Specifies the initial frame for interleaved files (zero for noninterleaved files).
- **dwStreams**
 - Number of streams in the file. For example, a file with audio and video has two streams.
- **dwSuggestedBufferSize**
 - Suggested buffer size for reading file. Should be large enough for largest chunk or record in file.
- **dwWidth**
 - Width of the AVI file in pixels.
- **dwHeight**
 - Height of the AVI file in pixels.

Stream header - 1

```
typedef struct {  
    FOURCC fccType;  
    FOURCC fccHandler;  
    DWORD  dwFlags;  
    DWORD  dwPriority;  
    DWORD  dwInitialFrames;  
    DWORD  dwScale;  
    DWORD  dwRate;  
    DWORD  dwStart;  
    DWORD  dwLength;  
    DWORD  dwSuggestedBufferSize;  
    DWORD  dwQuality;  
    DWORD  dwSampleSize;  
    RECT   rcFrame;  
} AVIStreamHeader;
```

After each stream header, is a *stream format* chunk.

Video: same header as a BMP file

Audio: Same format as FormatChunkData in [wav.h](#) in lab 3

Stream header - 2

- **fccType:** Type of the data in the stream: video, audio, text.
- **fccHandler:** Optionally, specifies installable compressor or decompressor.
- **dwPriority:** For example, relevant in a file with multiple audio streams.
- **dwInitialFrames**
 - How far audio data is skewed ahead of the video frames in interleaved files.
- **dwScale, dwRate**
 - Time scale for stream. $\text{dwRate} / \text{dwScale} = \text{number of samples per second}$. For video streams, this rate should be the frame rate. For PCM audio, the sample rate.
- **dwStart**
 - Starting time of file. For units, see **dwRate & dwScale**. Can specify a delay time
- **dwLength:** Length of this stream. For units, see **dwRate & dwScale**.
- **dwSuggestedBufferSize**
 - Size of buffer to read this stream. Typically, size of largest chunk in the stream.
- **dwQuality**
 - Measure of quality of data in stream. For compressed data, this typically represents the value of quality parameter passed to compressor.
- **dwSampleSize:** Size of a single sample of data. Set to zero if the samples can vary in size.
- **rcFrame**
 - Used in support of multiple video streams. Units for this member are pixels.
 - Destination rectangle for a text or video stream within the movie rectangle specified by the **dwWidth** and **dwHeight** members of the AVI main header structure.

Data Chunks

- Audio data
 - In AVI file as: (## is the stream identifier)
 - ‘##wb’
 - <frame data bytes>
- Video data
 - Either uncompressed
 - ‘##db’
 - <frame data bytes>
 - or compressed
 - ‘##dc’
 - <frame data bytes>

MPEG Standards

- MPEG-1:
 - Parts: Systems, Video, Audio
 - Applications include VCD (Video CD)
- MPEG-2
 - Parts: Systems, Video, Audio, Conformance, Reference Software Digital Storage Media Command and Control (DSM-CC), Advanced Audio Coding (AAC), Real Time Interface
 - Applications include MP3, DVD
- MPEG-3 (See MPEG-2)
- MPEG-4
 - Proposed fixed & mobile web standard
- MPEG-7: Multimedia Content Description
- MPEG-21: Multimedia Framework

MPEG: Audio

- MPEG standards have *layers*: Operating modes with increasing complexity and performance
- MPEG-1 provides
 - Mono and stereo coding at 32, 44.1, and 48 kHz sampling rate
 - Layer I, II, III: varying bit rates from 32 to 448 kbit/s
 - Optimized for 128 kbits/s stereo
- MPEG-2 BC (Backwards compatible)
 - multichannel extension to MPEG-1; ≥ 5 channels
 - Bit rate range: up to 1 Mbit/s
 - Some lower sampling rates 16, 22.05, and 24 kHz for bitrates from 32 to 256 kbit/s (Layer I) and from 8 to 160 kbit/s (Layer II & Layer III).
- MPEG-2 Advanced Audio Coding (AAC)
 - High-quality audio coding standard for 1 to 48 channels
 - Sampling rates of 8 to 96 kHz
 - Multichannel, multilingual, and multiprogram capabilities
 - Bit rates from 8 kbit/s to more than 160 kbit/s/channel
 - Multiple encode/decode cycles
 - Three profiles (layers) of AAC provide varying levels of complexity and scalability.