Week 7 – Standard MIDI Files

Roger B. Dannenberg
Professor of Computer Science and Art & Music
Carnegie Mellon University

Standard MIDI Files

→Key point: Must encode timing information←

Interleave time differences with MIDI data...

<track data> = 1 or more <track event>,

<track event> = <delta time> <event>,

<event> = midi data or <meta event>,

<meta event> = FF<type><length><data>

Delta times use variable length encoding.

Variable Length Coding

- 0-127 (7 bits) encoded in 1 byte
- If byte has high-order bit set, use low-order 7 bits PLUS what follows
- 0x81 0x00 => 1,0 => 0b10000000 => 0x80
- Largest number is FF FF FF 7F, which represents
  1111111 1111111 1111111 1111111
  (4 x 7 ones) = 0x0FFFFFFF.

Meta Events

- All start with FF – the MIDI reset command (so you cannot encode MIDI reset in SMF)
- Sequence Number
- Text “Events”: copyright, title, composer, lyric, cue point, rehearsal letters
- Tempo (microsec / quarter note)
- Time Signature
- Key Signature
- Sequencer-specific data

Standard MIDI Files – Review (2)

- `MThd` <length> <header data>
- `MTrk` <length> <track data>
- `MTrk` <length> <track data>

header info

track data: each with 16 channels

MIDI Files as Music Representation

- MIDI messages are limited to performance information
- Standard MIDI Files are somewhere between performance and symbolic notation, providing:
  - Time Signature
  - Tempo
  - Key Signature
- If timing is quantized to beats, you can recover a lot of notation:
  - Bar lines
  - Time signatures
  - Tempo and tempo changes
  - Keys and key changes

Limitations of Standard MIDI Files

- Parameters outside of MIDI:
  - Think of OSC data types and name space
  - Per-note parameters (MIDI control change messages affect entire channel)
- Accidentals (A ♭ or G♯ ?)
- Staves and Voices
  - But every track can have name
- Beams, Slurs, Clef
- Articulation Markings, Dynamics
  - Essentially all annotations
- Graphics
  - Stem direction
  - Spacing
- MIDI *does* have: tempo, time signature, key signature

SMF and Serpent

- Lots of code in serpent/programs/*.srp
- Allegro – an ascii music representation we'll talk about later
- allegro.srp has an *internal* representation:
- Alg_seq – a sequence object
  - Contains array of tracks
    - Each track contains array of events
  - Time map – piecewise-linear map time->beat
  - Time signature list

SMF and Serpent (2)

- Alg_event – each track is a list of these
  - has time and channel
- Subclassed into:
  - Alg_note – has pitch, key number, duration, loudness, parameters
    - Parameters are dictionary of attribute/value pairs
  - Alg_update – has key number, attribute, value

SMF and Serpent (3)

- mfread.srp – to parse SMF and build internal representation
- allegro_smf_read(file) returns an Alg_seq
- Mfwrite.srp – to output internal representation as SMF
- allegro_smf_write(seq, file) writes seq to a previously opened binary file (open(name, “wb”))
- seq2midi.srp – plays Alg_seq to MIDI port
  - See comments in the file & s2m_test()

SMF and Serpent (4)

- See also mftest.srp which shows how to get "pretty raw" data from SMF using Serpent's built-in SMF parser (written in C)
- This just keep a running total of delta times and calls a method each time an event is parsed, passing in parameters from the event.

Summary

- SMF is little more than MIDI messages (in binary form) interspersed with delta times in a variable length format
- Tracks were added to support sequencers
- Meta-data also added to support sequencers