MIDI literature review

MIDI stands for Musical Instrument Digital Interface and is used to compile all the information needed to reproduce a song under a common format. MIDI files allow specification of which note to play, for how long and with what velocity the instrument key is to be pressed [1],[4]. MIDI has become a standard interface used in the music industry and, for the sake of simplicity, will be the only format used in the project. MIDI files for a vast number of songs can be easily found online, allowing the team to easily choose any song available.

Each MIDI file starts with a *Header Chunk*. The header chunk specifies various parameters needed to decode the rest of the file. It is constructed as follow [2],[3],[5]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Length** | *4 bytes* | *4 bytes* | *6 bytes* | | |
| **Value** | *‘MThd’* | *Length (6bytes)* | *FF FF* | *NN NN* | *TT TT* |

Figure 1 MIDI Header Chunk

The top row in figure 1 indicates the size allocated to each section of the header. The bottom row is the actual MIDI header.

* The first 4 bytes of the header chunk correspond to **‘MThd’** in ASCII (4D 54 68 64 in HEX), this specifies that the incoming chunk is a header chunk.
* The next 4 bytes specify the length of the data which is to come. For header chunks this is always 6.
* The next 6 bytes of data specify three things [2],[3],[5]:
  + **FF FF -** The file format of the MIDI file. Three formats are accepted:
    - FF FF = 1: One track in the MIDI file.
    - FF FF = 2: Multiple synchronous tracks in the MIDI file. Which means that the file contains multiple tracks which all start at the same time.
    - FF FF = 3: Multiple asynchronous track in the MIDI file. Which means that the file contains multiple tracks all with different starting points
  + **NN NN -** The number of tracks in the MIDI file.
  + **TT TT -** The timing parameter The MIDI standard defines timing in its own way, which is *delta ticks per quarter note* and essentially defines the timing for the rest of the MIDI file

The remaining MIDI file consists of the *Track Chunk* Figure 2 shows its layout.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Length** | 4 bytes | 4 bytes | Variable depending on specified length | |
| **Value** | *‘MTrk’* | *Length* | *Delta time* | *Event* |

Figure 9 MIDI Track Chunk

* The first 4 bytes of the track chunk are **‘MTrk’** denoted in ASCII (4D 54 72 6B in Hex) this specifies that the incoming chunk is a track chunk.
* The next 4 bytes specify the length of the data which is to follow, which varies depending on the data.
* The following 4 bytes specify the *Delta Time,* this is the amount of time that needs to pass prior to executing the MIDI event.
* The last section of the header specifies the event which is to be executed.

|  |  |  |  |
| --- | --- | --- | --- |
| **Length** | 1 byte | 1 byte | 1 byte |
| **Function** | *Status Byte* | *Note Number* | *Note Velocity* |

The track event can be one of three things. Midi, meta and sysex events, all of which are always preceded by specific timing information. For this project, only MIDI events need to be considered. Midi events contain messages sent to each individual channel (instrument), an example Midi event message is shown in Figure 3. Midi events consist of three bytes, the first byte is known as the status byte and its function is to specify the type of command. A table with examples status bytes and their function can be found in figure 4. The second byte specifies the note to which the status byte is applied to and the third byte specifies the note velocity (loudness or softness). A table showing the musical note to which each binary value corresponds to can be found on **Appendix X**. [2][3][5]

*Figure 3: MIDI event message*

|  |  |
| --- | --- |
| Status Byte | Function |
| 0x80 | Channel 1 note off |
| 0x81 | Channel 2 note off |
| 0x90 | Channel 1 note on |
| 0x91 | Channel 2 note on |
| 0xB0 | Channel 1 Control Mode Change |

*Figure 4: MIDI Status Byte Example Functions*

A complete MIDI track chunk containing only MIDI events would look like the following:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Length** | 4 bytes | 4 bytes | Variable depending on specified length | | | |
| **Value** | *‘MTrk’* | *Length* | *Delta time* | *Status Byte* | *Note Number* | *Note Velocity* |

[1] <https://www.midi.org/specifications/item/the-midi-1-0-specification>

[2] http://www.petesqbsite.com/sections/express/issue18/midifilespart1.html

[3] http://www.indiana.edu/~emusic/361/midi.htm

[4] <https://www.midi.org/articles/an-intro-to-midi>

[5] <https://www.csie.ntu.edu.tw/~r92092/ref/midi/>

[6] <https://mido.readthedocs.io/en/latest/>

[7] https://docs.python.org/3.6/library/re.html