

MIDI's Role in Robot Rock

What is this document?

This document describes the pros and cons of MIDI as it applies to the Robot Rock project. My aim is not to cover the technical details of the MIDI specification and I'll refer you to other sources should you need more information. A brief overview is required because it is critical that we understand the fundamentals before we agree its inclusion into the project.

Overview

MIDI came about in the early 1980s as a solution to bridge the digital/music gap. As such, it should not be a surprise that MIDI stands for Musical Instrument Digital Interface. That's essentially what MIDI is: a means for controlling an instrument/device through digital signals.

It is vital to understand that the communication between MIDI enabled devices are sent as control signals **not** as audio samples. To be explicit, this means that when middle-c is pressed on a piano keyboard, a message is sent across to the MIDI receiver indicating that middle-c was pressed and with what velocity the key was struck with. The sound of middle-c, should there be one, would either be generated by the keyboard or the receiver would process the key-press event in one way or another.

The MIDI specification was certainly the first of its kind. It detailed the structure of the communication stream, defined a physical interface for instruments, and defined a file format for storing data. This was an incredible success, particularly because MIDI is still used heavily today; plenty of software exists/is produced and nearly every digital instrument still provides the physical interface. The music program on the UW campus offers a class on MIDI due to its establishment in the industry. Interestingly, because MIDI data essentially is a stream for real-time controls, many stage shows take advantage of the extensibility it offers by incorporating its usage to sync lighting and effect cues.

Despite all the wonder, the MIDI 1.0 specification is a product of its time. The word size of data traveling in a MIDI stream is 8 bits. As such, many boundaries of the specification are built around this limitation. Perhaps the most noticeable result of this from the user viewpoint is that a MIDI stream may refer to a maximum of 16 instrument channels. There are some other interesting issues that arise, but these are only apparent when you program the MIDI data stream directly.

To assist in the adoption of MIDI for the rapidly growing home PC user, the MIDI Manufacturers Association developed "General MIDI" in 1991. This is basically a standardized instrument kit that supports interoperability across platforms; for example, program change (see below) #3 will always be a grand piano under this contract. This was responsible for MIDI to become a popular choice for music sources in PC games as well as enabling rapid development possibilities for composers.

What information is in a MIDI stream?

A MIDI stream is not continuous; rather it sends packets of information. Like an Ethernet

network, the packets are tagged for specific devices and a device only responds to packets that match its corresponding channel number. Depending on the connection and behavior of the devices, the data may continue on to the next daisy-chained device and so on. (“Device” is a general term—they may be physical or exist as software.)

The stream contains global and individual device control information for up to 16 channels. Each channel may receive a number of cues:

- Note on/off – there are 128 possible notes, each corresponding to notes on the western scale
- Polyphonic aftertouch – post note-on pressure control
- Pitch wheel – to bend note pitch after a note-on event
- Program change – change a channel's instrument or strategy
- Control/Mode changes – the bulk to MIDI communication messages
 - Modulation wheel – for various effects
 - Breath control – controls for woodwind and brass instruments
 - Foot controllers – typically for keyboards
 - many others, including general purpose controls for user defined controls

Furthermore, the MIDI stream contains data that is common to all channels, such as timing and song information.

There are also meta-events for additional cues such as song lyrics (karaoke), author and copyright info, etc; as well as 'system exclusive' messages which are designed allow other, device specific messages via the MIDI stream such as, for example, transferring patches to a synthesizer.

Trying to support all of this data may seem like a daunting task; fortunately, not all of this information is needed for any given application and unsupported events may be easily bypassed. MIDI is used in many ways and all these messages exist to support the various ways people control music through digital means.

Integration with Robot Rock

First off, Robot Rock need not use MIDI internally. It is worth knowing, however, because at some level the AI are going to need to “see” what other AI unit are doing, and the MIDI model provides very heavy clues on what sort of information should be available in describing music.

Furthermore, a sound producing module is vital in Robot Rock. Many software synthesizer packages exists and almost all support the MIDI interface. The benefit is twofold: first, the sound producing component of the project is available; this reduces the scope of the project and fulfills the project requirement of using existing libraries; second, we may swap sound processors should we ever discover a need to. This is similar to the ADT/data-structure relationship, where the interface to the implementation is MIDI.

General MIDI will provide a pool of instruments for AI to choose from. This reduces the need to create an instrument list, find samples, an so on.

Having a popular interface/music description model will allow those familiar with it to begin creating AI much quicker. Also, having music data available as a higher abstraction as notes as opposed to just frequency/amplitude simplifies data interpretation for AI.

Pitfalls of Integration with Robot Rock

By using MIDI, we are initially limited by the MIDI specification. This includes being limited to 16 channels. This may be a boon, however, because having a limit early on lets us know how cluttered our interface may be.

With a pure GM powered sound processing unit, we are limited by the GM instruments. This removes the possibility of other fun-to-control, simple-to-implement instruments such as those that are Theremin-like. There is the possibility of folding the MIDI sound generator into a more general sound processor, one that allows playback of other methods (such as a wavetable synthesizer for the aforementioned Theremin-like instruments), but this expands the scope of the project.

References

- MIDI @ Wipedia <<http://en.wikipedia.org/wiki/Midi>>
 - Naturally a good source of general information about the topic
- MIDI Messages <<http://www.midi.org/techspecs/midimessages.php>>
 - Straight from the source
 - Sadly, the MIDI Manufacturers Association does not make their specification available on-line, but rather charges \$56 (!) for a printed manual. Fortunately, you may easily find information.
- MIDI handout @ Stanford's Center for Assisted Research in the Humanities <<http://253.ccarh.org/handout/smf/>>
 - A concise document on the details of the MIDI file format