Testing MIDI applications

Sema Kachalo

Jazz-Soft
USA

sema@jazz-soft.net

ABSTRACT

Testing is an important part of software development process. MIDI and Web MIDI applications are no exception. We introduce three open source tools to facilitate testing MIDI applications.

1. INTRODUCTION

Comprehensive testing of MIDI applications can be difficult. It may require MIDI equipment that is not always available. Web MIDI [1] applications require browsers that are challenging to automate. There are many corner cases that often remain untested such as: - user did not allow Web MIDI; - user allowed Web MIDI, but did not allow SysEx; - MIDI port is visible, but taken by another application; etc. We propose some tools that help to resolve these problems.

2. WEB-MIDI-TEST

WEB-MIDI-TEST [2] is a fake implementation of the Web MIDI API written in JavaScript. It does not require browser and can be used for automated testing. Its front end is identical to the Web MIDI API, and its back end allows test developer to create MIDI ports, emit and receive MIDI messages, change the port status to connected / disconnected / busy, and simulate different states of Web MIDI.

Typical test code would look like the following:

```
const WMT = require('web-midi-test');
const navigator = {
  requestMIDIAccess: WMT.requestMIDIAccess
};

const dest = new WMT.MidiDst('VIRTUAL MIDI-Out');
dest.receive = function(msg) {
  console.log('received:', msg);
};
dest.connect();

const src = new WMT.MidiSrc('VIRTUAL MIDI-In');
src.connect();
src.emit([0x90, 0x40, 0x7f]);

// ...
```

WEB-MIDI-TEST is perfect for unit testing, however our GitHub repository provides some examples of testing the entire page with Zombie.js [3], JSDOM [4], and Puppeteer [5].

3. MIDI-TEST

MIDI-TEST [6] creates virtual MIDI ports visible by the real MIDI system, and therefore, it can be used to test any MIDI applications, not necessarily written in JavaScript. It shares the same back end API with the WEB-MIDI-TEST. This allows to run certain tests in both flavors. Combined with JZZ [7], it can simulate Web MIDI API as well.

Typical test code looks almost identical to the previous example:

```
const MT = require('web-midi-test');

const dest = new MT.MidiDst('VIRTUAL MIDI-Out');
dest.receive = function(msg) {
  console.log('received:', msg);
};
dest.connect();

const src = new MT.MidiSrc('VIRTUAL MIDI-In');
src.connect();
src.emit([0x90, 0x40, 0x7f]);

// ...
```

At the moment, MIDI test works on MacOS and Linux.

4. TEST-MIDI-FILES

TEST-MIDI-FILES [8] is the framework that allows producing MIDI files from readable JavaScript code. It also provides some sample MIDI files along with the code. MIDI files can be used for interactive testing, or using JZZ-MIDI-SMF library [9], directed to the MIDI-TEST and WEB-MIDI-TEST scripts.

5. CONCLUSION

We have introduced three MIDI testing tools that are already successfully used by open-ource projects such as JZZ (no surprize:), WebMidi.js [10], and webaudio-tinysynth [11].

6. ACKNOWLEDGMENTS

We would like to thank the Web Audio Conference 2021 Committee for this great opportunity to present this work. We also thank all previous Web Audio Conference Committees for building a fantastic WAC community.

7. REFERENCES

[1] Web MIDI API: https://webaudio.github.io/web-midi-api

[2] web-midi-test: https://github.com/jazz-soft/web-midi-test

[3] Zombie.js: https://github.com/assaf/zombie

[4] JSDOM: https://github.com/jsdom/jsdom

[5] Puppeteer: https://github.com/puppeteer/puppeteer

[6] midi-test: https://github.com/jazz-soft/midi-test

[7] JZZ: https://github.com/jazz-soft/JZZ

[8] test-midi-files: https://github.com/jazz-soft/test-midi-files

[9] JZZ.midi.SMF: https://github.com/jazz-soft/JZZ-midi-SMF

[10] WebMidi.js: https://github.com/djipco/webmidi

[11] webaudio-tinysynth: https://github.com/g200kg/webaudio-tinysynth

Web Audio Conference WAC-2021, July 5–7, 2021, Barcelona, Spain.