Programming virtual musical instruments and audio effects in the Web browser

Michel Buffa, Jerome Lebrun, Jari Kleimola, Oliver Larkin, Stephane Letz

Programming virtual musical instruments and audio effects in the Web browser

Michel Buffa, Jérôme Lebrun
Université Côte d'Azur
CNRS, INRIA
(buffa, lebrun)@i3s.unice.fr

Jari Kleimola, Oliver Larkin
webaudiomodules.org
(jari, oli)@webaudiomodules.org

Stéphane Letz
GRAME
letz@grame.fr

ABSTRACT

WebAudio is a recent W3C API that brings the world of computer music applications into the browser. While JavaScript and Web standards are increasingly flexible and powerful, C/C++ has been the language of choice for real-time audio applications and domain specific languages such as FAUST facilitate rapid development with high performance. We present here a host Web application -the Pedal Board project- as well as plugins (instruments, effects), written in JavaScript, ported from C/C++, or written using Domain Specific Languages such as FAUST, and compiled to the new WebAssembly binary standard that can be executed by recent Web browsers.

These plugins follow an open standard we are developing (details on [4]).

1 - INTRODUCTION

This paper presents the actual works conducted separately by three groups of researchers who had different initial interests, but converged toward the notion of interoperable WebAudio plugins and decided to join forces to work towards an open standard.

Figure 1: the Zita_rev1 reverb written in FAUST

One group has been developing the FAUST [5] DSP domain specific language since 2002. Hundreds of music instruments and audio effects coded in FAUST are available, and the FAUST tool chain can compile them to different targets, including WebAudio. Figure 1 shows an example of audio effect (a complex reverberation) written in FAUST.

Another group has been developing Web Audio Modules (WAMs) since 2014 [3]. WAMs are high level audio plugins for the Web browser and have a C++ API, like native plugin formats. The WAM team ported to Web Audio famous commercial synthesizers such as the Yamaha DX7 or the Oberheim OBXd. See Figure 2.

Figure 2: online WAM instruments

The final group has been developing WebAudio plugins since 2012, in particular a guitar tube amplifier simulation and a pedal board for guitarists (see Figure 3). Each pedal, as well as the amp simulator is a plugin. Some authors of this paper are also W3C Advisory Committee representatives who participate in the W3C WebAudio working group.

All three groups had to deal with the concept of a WebAudio plugin, and decided to work together to make their plugins inter-operable. Furthermore, online demos are actively being developed as proof of concepts[1].

[1] Try the pedalboard host that works with the plugins made by the three different groups: https://wasabi.i3s.unice.fr/pedalboard also in this video: https://youtu.be/elbjh6tBK6U

2 - WEB-AWARE AUDIO PLUGINS

For the WASABI project [6], the WIMMICS team from INRIA/I3S/CNRS developed as a plugin the first online digital emulation of a real tube guitar amplifier: the Marshall JCM 800, a popular amp used by many classic rock artists (AC/DC, Led Zeppelin, Guns and Roses etc.) [1, 2], as well as a "host" application for assembling plugins together. The Pedal Board project is this "host" Web application that enable the creation of "virtual pedal boards" similar to the ones guitarists use. The set of plugins developed is pure JavaScript and rely exclusively on the Web Audio standard API.

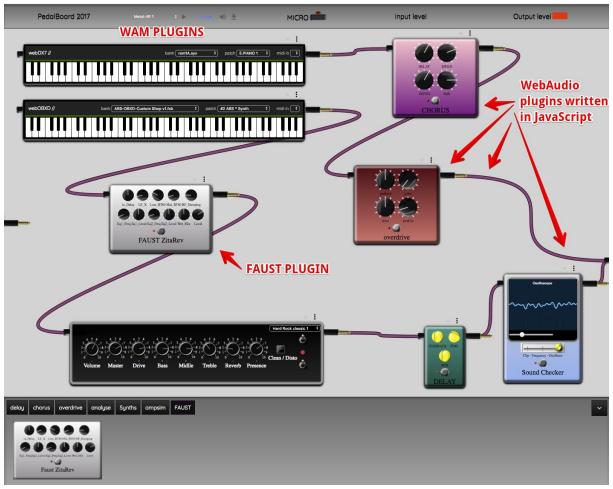

Figure 3: Two WAM synths and a FAUST plugin have been integrated in the pedal board

In order to develop an open Web Audio plugin standard, WIMMICS joined with the WAM and the FAUST teams to make their plugins inter operable in the pedal board. The final result is the pedal board application that shows how plugins from different origins / authors and written using different programming languages can be assembled and chained in order to mix MIDI controllable instruments and effects (Figure 3).

Although we aimed to introduce the functionality offered by the concept of native audio plugins and hosts to the Web, the differences of the environment require a different approach to the API design, and the development of a new API provided an opportunity to improve upon some aspects of native APIs. Advantages of web apps include: simple distribution (no installation, updates), collaboration (collaborative aspects can be implemented using WebSockets for example), platform independence, sandboxing (security). Disadvantages: efficiency (JavaScript is usually slower than native code, with issues that may affect real-time audio performance such as a garbage collector), latency (audio drivers on Linux/Windows), sandboxing (access to native resources, local hard disk access is forbidden or limited). An API should also be "Web aware" and use URLs as identifiers, should allow host webapps to discover remote plugins by querying plugin servers. Plugins should be usable without the need to download them manually, and the mixture of different JavaScript libraries and frameworks, should not raise any naming conflict or dependency problems. We also implemented all plugins as Web Components. This W3C standard[2] defines a way to easily distribute components with encapsulated HTML/CSS/JS/WASM code without namespace conflicts. Where native plugins needed to be downloaded and installed, URLs make no difference between a local or a distant plugin, a Web Component plugin could be used remotely just by its (possibly RESTful) URL reference. This makes writing a plugin remote server easy.

7 - CONCLUSION

This paper is the first initiative that involves synchronising the efforts of three groups of developers who have been working on various approaches for implementing high level audio "plugins" in the browser. We hope it may become a starting point for a new addition to the WebAudio v2 specification.

ACKNOWLEDGMENTS

ElMahdi Korfed and Guillaume Etevenard who helped developing the pedal board webapp. French Research National Agency (ANR) and the WASABI project team (contract ANR-16-CE23-0017- 01).

REFERENCES

[1] Buffa, M. & Lebrun, J. (2017, Aug). Real time tube guitar amplifier simulation using WebAudio. In *Proc. 3rd Web Audio Conference 2017 – Collaborative Audio #WAC2017, London, United Kingdom*.

[2] Buffa, M. & Lebrun, J. (2017, Aug). Web Audio Guitar Tube Amplifier vs Native Simulations. In *Proc. 3rd Web Audio Conference 2017 – Collaborative Audio #WAC2017, London, United Kingdom*. (http://wac.eecs.qmul.ac.uk/)

[3] Kleimola, J. & Larkin, O. (2015). Web audio modules. In *Proc. the Sound and Music Computing*, 2015.

[4] Buffa, M. & Lebrun, J., Kleimola J., Larkin O., Letz S.. Towards an open Web Audio plug-in standard. *WWW '18 Companion, Mar 2018, Lyon, France.*

[5] Orlarey, Y., Fober, D. & Letz, S. (2004). Syntactical and Semantical aspects of Faust. *Soft Computing, 8(9):623–632, 2004.*

[6] Buffa, M. & al. (2017, Aug.).WASABI: a Two Million Song Database Project with Audio and Cultural Metadata plus WebAudio enhanced Client Applications. In *Proc. 3rd Web Audio Conference 2017 – Collaborative Audio #WAC2017, London, United Kingdom*.

[2] https://www.webcomponents.org