Emerging W3C APIs opened up commercial opportunities for computer music applications

Michel Buffa, Jerome Lebrun, Shihong Ren, Stéphane Letz, Yann Orlarey, Romain Michon, Dominique Fober

Emerging W3C APIs opened up commercial opportunities for computer music applications

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

Shihong Ren, Stéphane Letz, Yann Orlarey, Romain Michon, Dominique Fober
GRAME, 11 cours de Verdun LYON
renshihong@hotmail.com
(letz, orlarey, michon, fober)@grame.fr

Introduction

In 2018, with a group of researchers and developers (some of whom are members of the W3C WebAudio WG) we proposed a WebAudio Plugin standard (WAP) and gave birth to a growing ecosystem for the development of computer music applications in the browser [1]. These plugins can be seen as a transposition of what exists in the native world, adapted to be web-aware (i.e. a plugin can be remotely included in any host web audio application by a URI). Since then, many contributions have been made and many plugins have been developed. Not only several "host" applications have appeared, including commercial ones, but also tools to help developers, including an online IDE coded as a Progressive Web App (PWA) to write and publish plugins from A to Z, the audio DSP core of the plugins being compiled into WebAssembly. Some of the plugins we developed during the French research project WASABI are now sold as add-ons to an online Digital Audio Workstation (Amped Studio), showing the new opportunities that these emerging APIs have created.

The emergence of WebAssembly for DSP algorithms

Plugins can be written 1) in pure JavaScript, only using the high-level features of the WebAudio API, or 2) can be written in C/C++ or using DSLs and cross-compiled to WebAssembly. Rapidly, it has become important to provide tools to facilitate the development of DSPs that can be compiled to this new standard. Many developers used to code native plugins in C or C++ have gotten into the habit of prototyping the DSP part with DSLs like FAUST [6]. This is why we recently designed a new IDE for this language, which takes advantage not only of WebAssembly (the FAUST compiler itself was compiled in WebAssembly [7]), but also of other standards emerging from the W3C (PWA, WebMIDI, WebAudio, WebComponents, etc.).

An online IDE for RAD development of WebAssembly-based WebAudio Plugins

This online IDE facilitates the development of audio plugins to be deployed on different targets (different native plugin formats are supported) but we will focus here on the production of WebAudio plugins [2]. The IDE embeds a multi-file source code editor that supports IntelliSense-like features such as syntax highlighting and auto-completion. It also includes the WebAssembly version of the FAUST compiler, enabling developers to write, compile, run and test their code directly in the browser thanks to the WebAudio API. This IDE supports multiple audio inputs (physical audio devices, MIDI interfaces, computer keyboard), different types of signal visualizations (spectrogram, oscilloscope, etc.), and generates a default GUI for testing the DSP coded (the FAUST language supports a set of primitives for describing abstract GUIs), as shown in Fig 1.

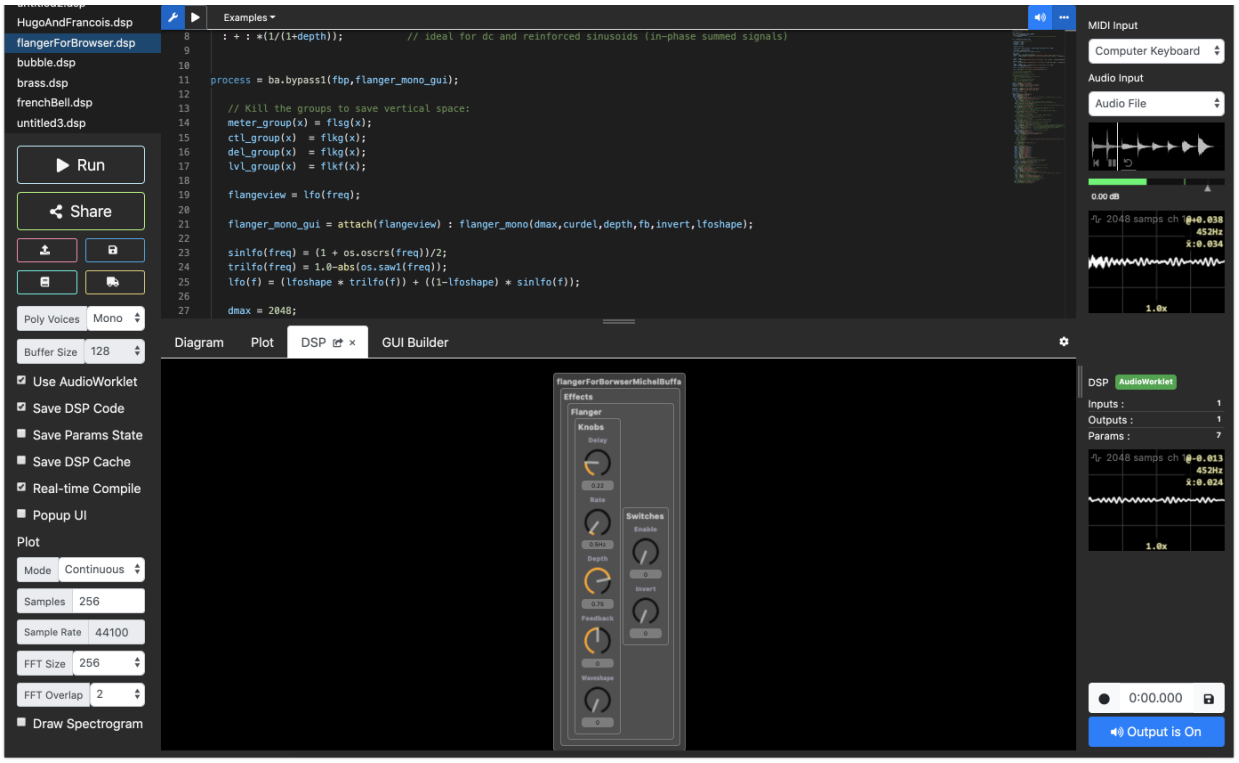

Fig 1: the online FAUST IDE showing a default GUI for testing the compiled code.

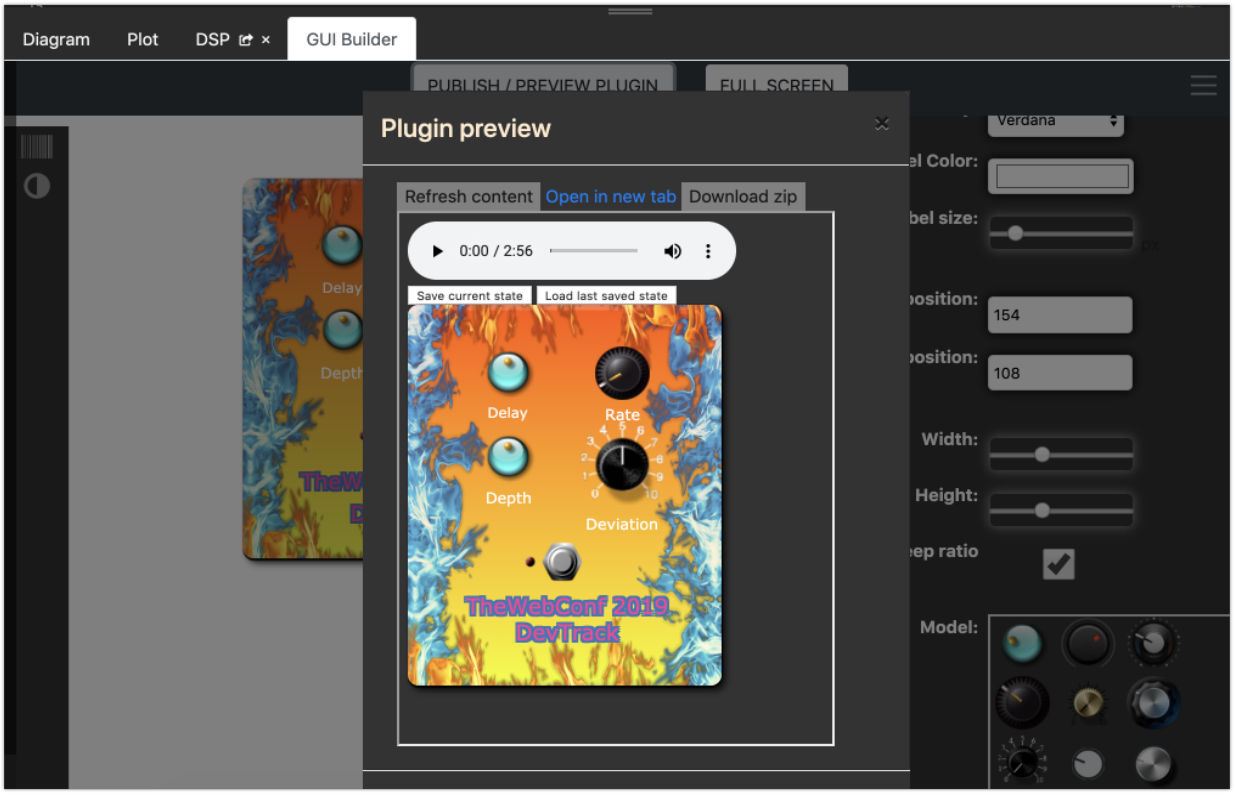

Fig 2. A GUI builder is embedded in the IDE. Plugins can be tested online with their custom built GUI, downloaded as local zip files, published on remote WebAudio Plugins repositories.

PWA, GUI Builder and publishing to online remote plugin servers

Finally, the IDE embeds a GUI builder based on WebComponents (Fig 2.). One can develop an audio plugin from scratch (or reuse code of existing ones), including a customized GUI, test it, download it as an archive (a zip file) or deploy it to a remote WAP server so that host applications can "discover" the published plugins automatically (Fig 4.). Plugins are all running WebAssembly code and the final GUI is based on WebComponents. The IDE is a PWA and can be used offline with some limitations (i.e no remote publishing of plugins).

Plugins and host applications developed so far...

More than 70 plugins have been developed so far by multiple contributors. A subset can be seen in Figures 3 and 4.

- Guitar tuner, equalizers, mixers,
- Tube guitar amplifier simulations [4, 5],
- Many audio effects : delay, flanger, chorus, phasers, wah wah, stereo enhancers, several types of non linear distortions (recreations of famous FX pedals for guitarists such as the Big Muff by Electro-Harmonix, the Roland TS9 overdrive, ...), and many more to come.
- Several instruments: analog synthesizers [8] (recreations of the Oberheim OBxD, Yamaha DX7, Korg Minilogue, etc.), instruments based on physical modeling (djembé, guitar)...

Host web applications include:

- The Virtual Pedalboard app for assembling plugins in a graph of audio effects and instruments [3], shown in Fig 3, is a virtual recreation of the pedalboards that guitarists use on stage. These are composed of a set of audio FX pedals connected together, the output of the pedalboard going to a guitar amplifier.
- A meta plugin called a "Rack" that is "a plugin for assembling plugins" that acts as a host and comes with multiple banks and presets to produce famous guitar sounds, inspired by the "guitar pedal and amp" manager of the GarageBand DAW application available on Apple devices (Fig 4.)
- Digital Audio Workstations (DAWs, see Fig 5.) for recording, editing and mixing multitrack songs, each track being associated with one or more plugins. For example, a guitar track will process the dry sound that has been recorded, through a set of audio effects and/or a guitar amplifier simulation. A keyboard track will record MIDI events that will be played by a virtual instrument. Each of these (effects/instruments) could be a WebAudio Plugin. During the course of the WASABI ANR French research project, we developed more than 70 open source plugins, some of them being now commercialized by the I3S laboratory (part of French CNRS).

All these tools are available online[1] and can be used for free (Amped Studio is a commercial application that comes with a free plan, with some limitations), most are open-source (links are provided at the end of this paper). Videos show some demonstrations of these tools[2] and a step by step tutorial for creating WebAudio plugins is also available online[3]. Note that: as of early 2020, you need an up-to-date browser based on

[1] Virtual Pedalboard: https://wasabi.i3s.unice.fr/dynamicPedalboard/#, Faust IDE with GUI Builder for WebAudio plugins: https://mainline.i3s.unice.fr/fausteditorweb/dist/, AmpedStudio: https://ampedstudio.com/.
[2] Videos: Virtual Pedalboard tutorial: https://www.youtube.com/watch?v=pe8zg8O-BFs, guitar amp sim: https://www.youtube.com/watch?v=lIfm9ZMtG-I, GUI Builder and plugin publication: https://youtu.be/do-9mA42pNA, also check all this YouTube channel: https://www.youtube.com/channel/UC0vZKRYdSHckcngkRMVyZRQ/videos
[3] https://tinyurl.com/r223p7c

Chromium, or Firefox Nightly to run these applications. They are the only browsers with the required API implementations, and polyfills for some of these APIs are not available or lack performance.

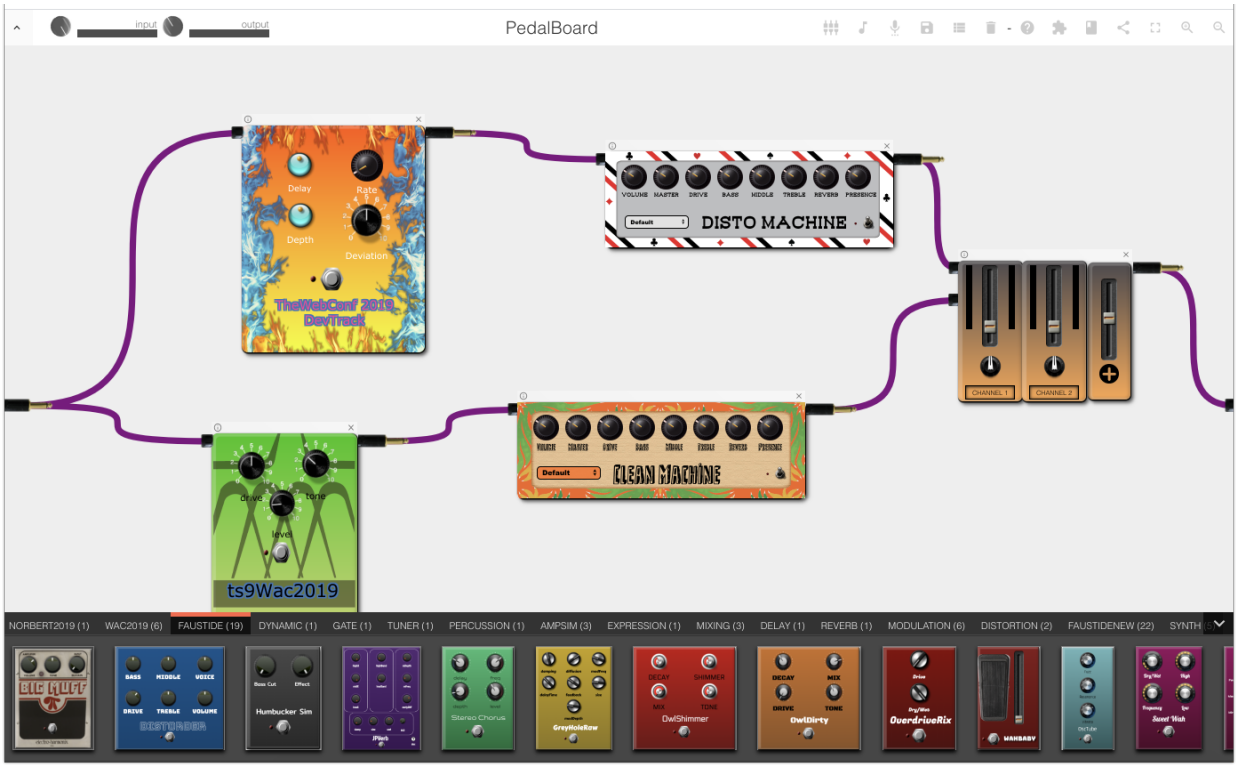

Fig 3. The Virtual Pedalboard host application discovers newly created plugins (using the REST APIs of WAP servers). We can see on top left the plugin from Fig. 2.

FIg 4. The “Rack” plugin. Inspired by GarageBand’s pedal FX and Amp Sim manager.

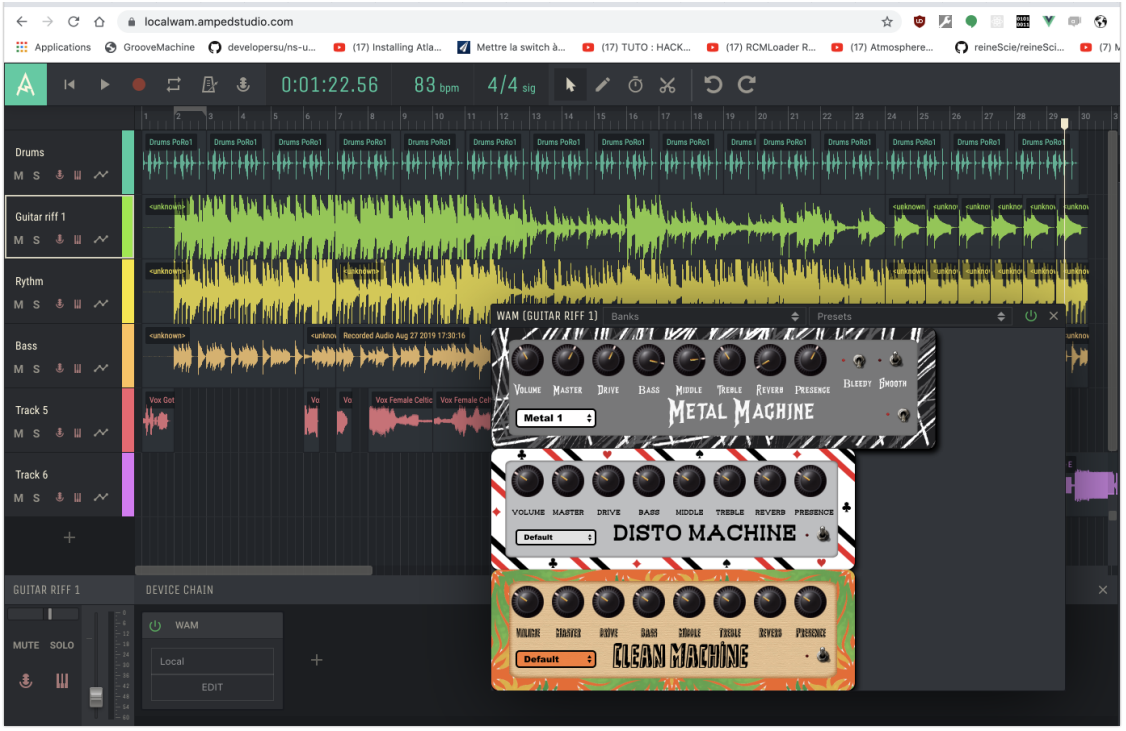

Fig 5. The Amped Studio online digital workstation (DAW) supports WAP plugins.

Performance and compatibility concerns

Not all of the needed APIs are equally available on all browsers. The AudioWorklet API, part of the WebAudio specification, required to run WebAssembly DSP code, is only publicly available on Chromium-based browsers (in this implementation, the choice not to have the audio thread with real-time priority can cause audio glitches in some circumstances, and is currently being questioned), while an experimental version is available in Firefox Nightly, this time with a real-time approach. The WebMIDI implementation is still only available in Chromium-based browsers for now. However, three commercial online Digital Audio Workstations have been online for more than one year now, running successfully on Chromium-based browsers, and will work on other browsers when implementations of the needed APIs will become available (polyfills are not always available).

Security concerns

When a host application uses dynamically loaded plugins, it is obvious that some security precautions need to be taken. The WAP standard comes with an SDK but also with online validation tools [1]. For example, developers can check that a plugin complies with the WAP specification, can check its main features, save/restore its state etc... Nevertheless, a publishing workflow is necessary before making a plugin available in a commercial online host application such as Amped Studio. Tools are available to test plugins locally on the developer's machine before submitting them to the internal validation process set up by host app maintainers (plugins can be tried using an http server running on localhost, and can be seen by the host webapp), then the plugin is submitted for further validation, including manual checks. The workflow is similar to that of publishing mobile applications to app stores: automatic tools and manual validation are involved.

Conclusion

During the presentation we propose to demo these tools and give details on the design principles that guided us so far. These WebAudio applications and tools show that online computer music applications can convincingly compete with their native counterparts, making easier some use cases such as collaborative music composition - as illustrated by SoundTrap being bought by Spotify and Amped Studio being offered to the 300 million users of NetEase Cloud Music[4] a Chinese music social network (sort of Deezer/Spotify in China). This shows the maturity reached by recent Web APIs such as WebAudio, WebMIDI, WebAssembly, WebComponents and others. And this also opened up commercial opportunities: open source WebAudio software developed in the framework of academic projects such as the French ANR WASABI project, (guitar amp simulators, FX pedals, Rack plugin) are now being marketed as addons for the Amped Studio DAW.

Acknowledgements

This work was supported by the French Research National Agency (ANR) and the WASABI team (contract ANR-16-CE23-0017-01).

Open source resources

- FAUST and FAUST IDE: https://faust.grame.fr/ https://faust.grame.fr/ide
- WAP SDK: https://github.com/micbuffa/WebAudioPlugins
- Virtual Pedalboard: https://github.com/guizmo2000/Wasabi-Pedalboard/tree/dynamicLoading
- Tube guitar amp simulation designer: https://mainline.i3s.unice.fr/AmpSim4/

Bibliography

[1] Buffa, M., Lebrun, J., Kleimola, J., Larkin, O. and Letz, S. “Towards an open Web Audio plugin standard”. In *Companion Proceedings (Developer’s track) of the The Web Conference 2018* (WWW 2018), Mar 2018, Lyon, France. ⟨hal-01721483⟩

[2] Shihong Ren, Stéphane Letz, Yann Orlarey, Romain Michon, Dominique Fober, et al.. “FAUST online IDE: dynamically compile and publish FAUST code as WebAudio Plugins”. *WebAudio Conference 2019*, Dec 2019, Trondheim, Norway. ⟨hal-02366725⟩

[3] Michel Buffa, Jerome Lebrun. “WebAudio Virtual Tube Guitar Amps and Pedal Board Design”. *WebAudio Conference 2018*, Sep 2018, Berlin, Germany. ⟨hal-01893781⟩

[4] Michel Buffa, Jerome Lebrun. “Real time tube guitar amplifier simulation using WebAudio”. *Web Audio Conference 2017 – Collaborative Audio* #WAC2017", Queen Mary University of London, Aug 2017, London, United Kingdom. ⟨hal-01589229⟩

[5] Michel Buffa, Jerome Lebrun. “Real-Time Emulation of a Marshall JCM 800 Guitar Tube Amplifier, Audio FX Pedals, in a Virtual Pedal Board”. *WWW2018 - TheWebConf 2018 : The Web Conference, 27th International World Wide Web Conference*, Apr 2018, Lyon, France. ⟨10.1145/3184558.3186973⟩. ⟨hal-01721463⟩

[6] Orlarey, Y., Fober, D., & Letz, S. 2004. “Syntactical and semantical aspects of Faust”. In Soft Computing, 8(9), 623-632.

[7] Letz, S., Orlarey, Y., and Fober, D. 2018. “Faust Domain Specific Audio DSP Language Compiler to WebAssembly”. In *Compagnion Proc ot the Web Conference, International World Wide Web Conferences Steering Committee, Lyon France 2018*.

[8] Kleimola, J. and Larkin, O. 2015. “Web Audio modules”. In *Proc. 12th Sound and Music Computing Conference* (SMC 2015), Maynooth, Ireland.

[4] https://en.wikipedia.org/wiki/NetEase_Music