Develop WebAudio Plugins in a Web Browser

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

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

ABSTRACT

We propose to demo an online IDE based around the FAUST DSP audio language [1], that includes a source code editor, embedded compiler and GUI editor allowing to directly test, generate and deploy WebAudio Plugins (WAP). The tool is available online[1].

1. INTRODUCTION

When audio effects or audio/MIDI instruments have to be shared between several DAWs or audio environments, a plugin model is usually preferred.

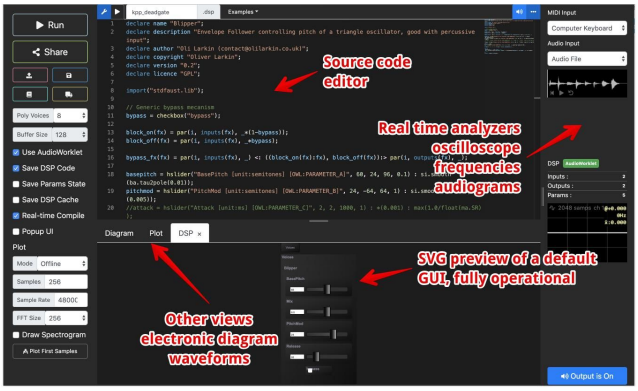

Figure 1: the FAUST IDE provides many embedded tools: oscilloscopes, spectroscope and spectrogram, functional default GUI, schema preview, etc.

Several native audio plugin formats are now popular, including Steinberg's VST format (Virtual Studio Technology, created in 1997 by Cubase creators), Apple's Audio Units format (Logic Audio, GarageBand), Avid's AAX format (ProTools creators) and the LV2 format from the Linux audio community. In the much newer WebAudio API (2011), there was no standard format for high-level audio plugins. With the emergence of Web-based audio software such as digital audio workstations (DAWs) developed by companies such as SoundTrap, BandLab or AmpedStudio, it is desirable to have a standard to make WebAudio instruments and effects interoperable as plugins compatible with these DAWs and more generally with any compatible host software.

Such a plugin standard needs to be flexible enough to support these different approaches, including the use of a variety of programming languages. New features made possible by the very nature of the Web platform (e.g., plugins can be remote or local and identified by URIs) should also be available for plugins written in different ways. To this end, some initiatives have been proposed [2, 3] and with other groups of researchers and developers we proposed [4] a standard for WebAudio plugins called WAP (WebAudio Plugins), which includes an API specification, an SDK, online plugin validation tools, and a series of plugin examples written in JavaScript but also with other languages[2]. These examples serve as proof of concept for developers and also illustrate the power of the Web platform: plugins can be discovered from remote repositories, dynamically uploaded to a host WebApp and instantiated, connected together etc. The reader can get a "multimedia" idea of this work by watching online videos that present the results of this work[3]. Since the last year, WAP now includes support for pure MIDI plugins (a GM midi synthesizer, virtual midi keyboards, a MIDI event monitoring plugin, etc). We propose to demo a new online IDE, that is well suited for coding, testing, publishing WAP plugins written in FAUST, directly in a Web browser. The IDE includes a GUI editor that allows developers to fine-tune the look and feel of the plugins. Once complete (DSP + GUI) the plugins are packaged in the form of standard W3C WebComponents and published on remote WAP plugin servers. The plugins will then be directly usable by any compatible host software, using their URIs.

2. THE ONLINE IDE

We embedded a WebAssembly version of the FAUST compiler in the IDE created by the Emscripten transpiler, to dynamically generate WebAudio nodes from FAUST DSP codes. The node

[1] https://faust.grame.fr/ide/, experimental version with the WAP GUI Builder available at https://mainline.i3s.unice.fr/idewap

[2] https://github.com/micbuffa/WebAudioPlugins
[3] https://www.youtube.com/watch?v=pe8zg8O-BFs

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: owner/author(s).
Web Audio Conference WAC-2019, December 4–6, 2019, Trondheim, Norway.
© 2019 Copyright held by the owner/author(s).

can be an *AudioWorkletProcessor* or a *ScriptProcessor* to be connected to audio I/O devices or other DSPs.

Based on this main feature, we built a code editor with full IDE user experience that could provide more information and details of a DSP through graphical representation in a Web page. A DSP developer probably not only needs to hear how the DSP sounds, but also to test it with other audio inputs, or to precisely display the time domain and frequency domain data of outputs. We have added several testing, visualisation and debugging tools into a basic code editor with following UI layout:

- All options related to FAUST code compilation are situated using controllers from the left sidebar panel, including a virtual file system manager that can be used by the compiler.
- All options and displays related to DSP runtime, such as MIDI, audio inputs, a recorder, and quick signal probing are placed in right sidebar panel.
- The remaining central region of the page is divided into two parts with configurable heights: a source code editor on the top and a multi-tab display panel which can display the logs from the compiler, a FAUST block diagram corresponding to the DSP code, a larger signal scope, a running GUI of the plugin being developed, and finally a GUI Builder / exporter for designing the user interface a WAP plugin version of the code, usable in external host applications.

In the signal scope panel, we designed four modes of signal visualisations: data table, oscilloscope (stacked and interleaved by channels), spectroscope and spectrogram to help FAUST users to debug their DSPs. After a FAUST DSP is tested in the editor, users can export the DSP to different architectures including WebAudio Plugins (WAPs). A dedicated GUI builder is integrated in the online IDE that receives FAUST DSP's GUI definitions while it is compiled.

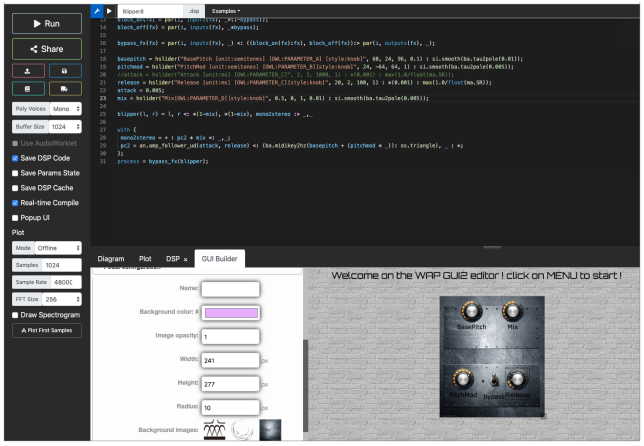

Figure 3: The default GUI can be customized: change textures, knobs, sliders, switches positions size, look and feel and labels etc.

At any time the plugin (DSP + GUI) can be tested from within the IDE, without the need to download it on a local disk. It is then possible to refine the GUI, adjust the layout, adjust the look and feel of the controllers among a rich set of knobs, sliders, switches (Figure 4 shows different looks and feels that can be created from the same DSP code). At any time, the plugin can be published on a remote plugin server, using standard Web services. The plugins will then be directly usable by any compatible host software such as PedalBoard[4].

3. ACKNOWLEDGMENTS

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

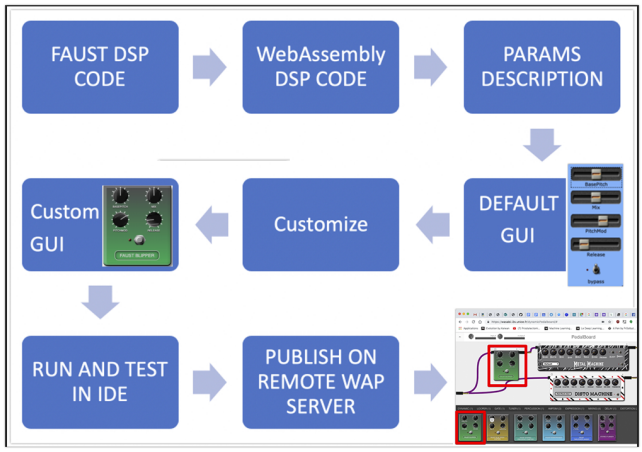

Figure 5: workflow of the end-to-end design and implementation of a WebAudio plugin

Figure 6: the virtual pedalboard host application scans multiple remote WAP plugin servers.

4. REFERENCES

[1] 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*.

[2] Jillings N. and al. 2017. "Intelligent audio plug-in framework for the Web Audio API". In *Proc. 3rd Web Audio Conference* (WAC 2017). London, UK.

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

[4] 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〉

[4] https://wasabi.i3s.unice.fr/dynamicPedalboard/