WebAudio Virtual Tube Guitar Amps and Pedal Board Design

Michel Buffa, Jerome Lebrun

WebAudio Virtual Tube Guitar Amps and Pedal Board Design

Michel Buffa
Université Côte d'Azur,
CNRS, INRIA
buffa@i3s.unice.fr

Jerome Lebrun
Université Côte d'Azur,
CNRS
lebrun@i3s.unice.fr

ABSTRACT

In this paper are exposed our latest experiments with the WebAudio API to design different types of gears for guitarists: real-time simulations of tube guitars amplifiers, fx pedals, and their integration in a virtual pedal board. We have studied different real guitar tube amps and created an interactive Web application for experimenting, validating and building different amp designs that can be run in browsers. Blind tests have been conducted with professional guitar players who compared positively our real-time, low-latency, realistic tube guitar amps simulations with state-of-the-art native equivalents. We also created a set of "virtual audio fx pedals" that implement popular audio effects such as flanger, chorus, overdrive, pitch shifter etc. These amps and pedals simulations can be packaged as "WebAudio plugins" and stored in plugin repositories (REST endpoints or local folders). We also developed a "host" application -a virtual pedal board- that allows us to scan repositories for plugins and to chain/assemble them[1].

1. INTRODUCTION

In some previous papers [1,2], we detailed the design of a classic tube guitar amp, the Marshall JCM 800, and showed how its different stages could be emulated using the WebAudio API. More recently, we also proposed with other researchers, an open plugin standard for WebAudio [3]. In Section 2, we will show our new results towards a framework for versatile and modular tube amp simulations. Instead of emulating a given guitar amp, we create a versatile amp designer to customize the typical parts that compose a guitar amp: preamp, tonestack, power amp, reverb, speaker and cabinet, and to assemble them with different topologies or adding filter banks between or inside these stages (Figure 1).

These modular tools make it possible to create specific amp designs targeted to get clean, crunch/bluesy, classic rock distortion, hi gain/metal sounds, etc. The final result can be a single specific amp or a multi channel amp that will use several switchable designs (e.g. a clean and a crunch channel). In Section 3, we will detail the WebAudio plugin open standard proposal we converged on with other researchers [3]. In this framework, the amp simulations, along with other audio effects can be packaged as reusable/interoperable plugins and published in repositories before being exploited by host applications. Section 4 will present our "virtual pedal board": a "host" application that discovers and loads dynamically plugins that can then be connected together in order to make richer sound designs.

[1]All these developments are open source and can be tried online at: https://wasabi.i3s.unice.fr/dev/

Web Audio Conference WAC-2018, September 19–21, 2018, Berlin, Germany.

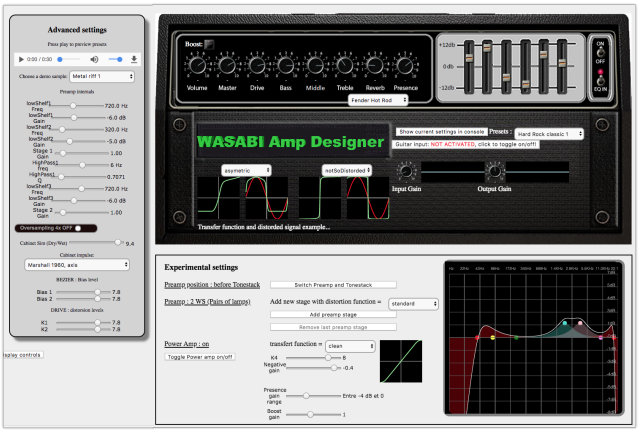

Figure 1. The amp designer GUI[2].

2. GUITAR TUBE AMP DESIGNER

Previously, we made a survey on tube guitar amplifier simulations, and detailed quite extensively the different stages of an iconic tube guitar amplifier, the 'Marshall JCM800', that we recreated using the WebAudio API [1,2]. In these implementations, the different stages were perceptually approximated using the basic tools available from the WebAudio API: mainly waveshapers, biquad filters and convolver nodes. Even though we had to face some caveats and limitations with this approach, we succeeded in recreating an enjoyable and pretty faithful simulation of the JCM800, as assessed by the measures (latency, frequency responses) and many focus group blind tests with professional guitar players but also presentations/demos conducted at different conferences [1,2,3]. To our knowledge, there is still no other academic work in this field from WebAudio researchers. Reference papers exist out of this community, but which do not take into account the constraints one has to face when doing real-time audio within a browser (only 128 sample accuracy, latency issues, thread priorities, etc.).

In this paper, we will focus on the major improvements from our previous simulations: how we recently tackle the critical stages that were previously overlooked/simplified, but also how it led to a major change of paradigm towards a fully versatile and modular framework for the recreation/design of vintage or new guitar tube amplifiers. Since a vast majority of the 'Golden Age' guitar tube amplifiers [4,5,6,7,8] (such as the ones from Vox, Marshall, Fender, Mesa-Boogie …) are relying on rather similar topologies and common stages, our description will be stage-by-stage:

[2] At: https://youtu.be/-NdMdJQx2Bw or https://wasabi.i3s.unice.fr/AmpDesigner

2.1 The Preamp Stage

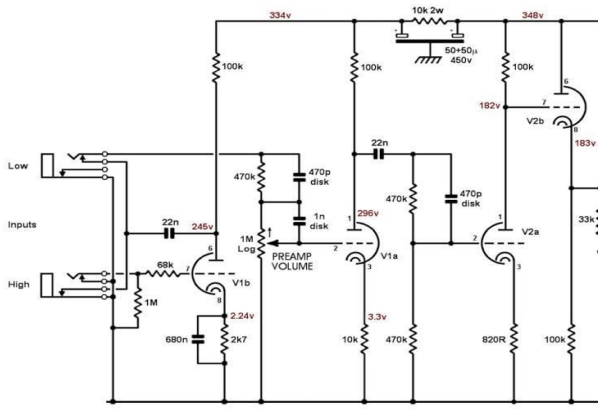

Figure 2. A typical preamp stage (Marshall JCM800).

The 'Preamp' stage whose major purpose is to transform/shape a high impedance, low voltage (hundreds of mV) signal coming from the guitar pickup into a low impedance, medium voltage (tens of V) output signal capable of driving a Power Amp. Figure 2 shows the JCM800 preamp with 4 tubes and filters between them, that we had recreated in our previous simulations [1,2].

First improvement - the number and type of tubes / associated filters is now customizable: our experiments with waveshapers and biquad filters on the preamp gave nice results in the past [1,2]. However, this time we added the possibility to interactively play with the number of cascaded tube stages (Figure 3) in the preamp, to modify the type of tube used (triode, beam-tetrode, pentode, ...) by changing the transfer curves of the 'waveshaper'[3]. Curves can be symmetric, asymmetric, clipped or even parametric Bezier style with biasing control [4,5,8]. We provide dozens of pre-configured curves corresponding to some existing tube models (Figure 4), and controls to adjust filters before and after each tube/waveshaper. As a general rule, more tubes means a thicker sound.

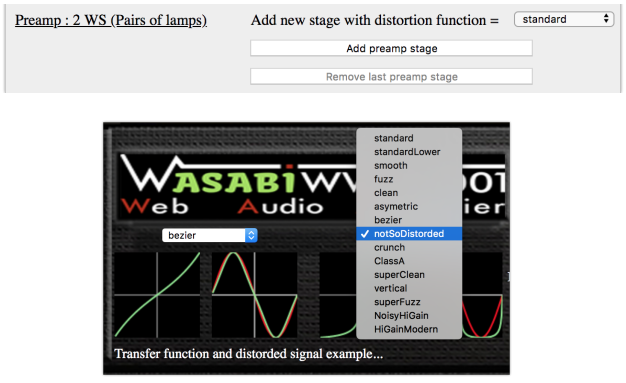

Figure 3. The number and types of tubes/filters in the preamp can be adjusted / tubes and filters can be customized.

Second improvement - introducing a tube transfer function editor: waveshaper curves can now be adjusted using mouse or fingers, even while playing, allowing instant feedback -you can hear your modifications in real time (which is interesting for pedagogy) and you can see the impact on the generated harmonics thanks to some real-time frequency analyzers we embedded in the GUI. These analyzers allow to probe the signal at different locations in the amp design. Based on Bezier curves, the interactive editor allows constrained bias adjustment (just click and drag the central control point to make the curve asymmetric), and offers many different shortcuts to 'squeeze' or 'squarify' the curves to stay within the range of realistic tube transfer functions [9].

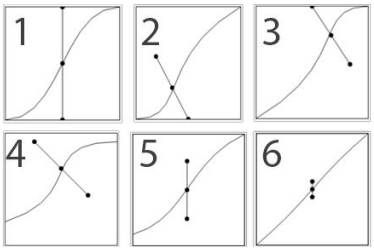

Figure 4. The amp designer embeds a transfer curve editor[4].

Third improvement - the amp designer also supports different preamps designs/topologies: in addition to the "classic rock" preamps illustrated by Figure 2 and 5, we also introduced a multiband preamp that gives radically different sound colors, more pristine/sparkling sounds that we were not able to achieve otherwise (Figure 6). Other topologies will be added in the future.

Figure 5. Serial preamp, WS + filters for each stage.

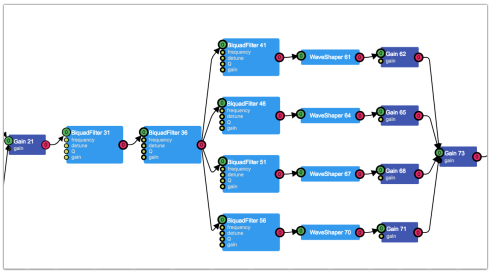

Figure 6. Multiband preamp (filter + WS in parallel) to limit inter-modulations.

Fourth improvement - the slope of the tube transfer function curves can now be driven real-time by the power of input signal enveloppe: improving the real-time dynamics in simulations is crucial to get them more realistic. Namely, when pushed to their limits by large transients, classic guitar tube amplifiers (esp. with tube rectifiers), unlike transistor ones, have a tendency towards harmonic saturation with dynamic range compression (known as 'sag') and temporal lag (known as 'squish')[5]. Sag is consecutive to some drooping of the supply voltage in the preamp stage in response to large transients. Now, squish is linked to some temporal hysteresis induced by lag in the feedback loop - also in response to large transients. These effects are quite tricky to emulate in real-time[6]. Consequently, to mimic this hysteresis and the sag in the response of a tube preamp stage, we implemented it with simple dynamical real-time changes in the slopes of the transfer functions in our preamp simulation. By properly adjusting the threshold at which we get the squish, only the higher power transients from the envelope amplitude will trigger controlled change of slope. This is still an experimental feature that needs

[3] Notes on Waveshaping: https://rhordijk.home.xs4all.nl/G2Pages/Distortion.htm

[4] Online demo: https://output.jsbin.com/zotaver

[5] More details on: http://www.aikenamps.com/index.php/what-is-sag

[6] Some examples at: http://www.diale.org/tube_emulation.html

to be properly adjusted and evaluated. Figure 7 shows a curve animated in real time with hysteresis behaviour in the transfer function triggered by the input signal envelope.

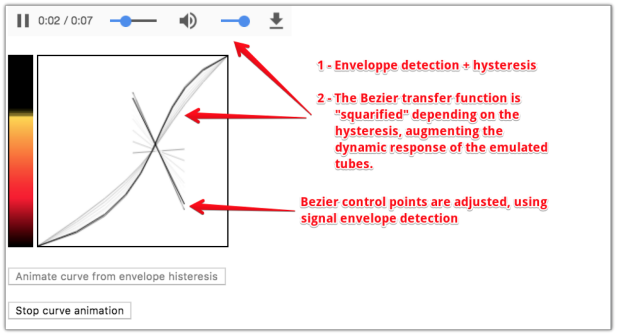

Figure 7. Boosting the dynamic response by hysteresis of the emulated tubes, using envelope power detection.

2.2 The Tonestack

In a reference paper about the modeling of the Fender Bassman amp tonestack [5,6], David Yeh and Julius Smith explained the importance of this circuit as it shapes by its controls (bass, medium, treble) the sound of an electric guitar. Unfortunately, it cannot be simulated accurately using filters in series or in parallel. Despite its apparent simplicity (Figure 8), this circuit is a complex filter network with each part impacting the others. Using the technique of virtual analog, Pakarinen and Yeh proposed a simulation faithful to the analog prototype [6,7].

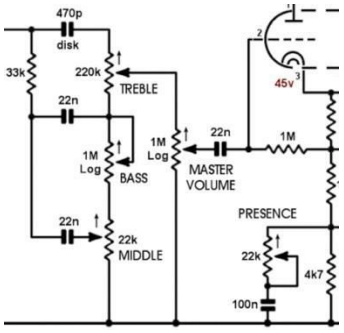

Figure 8. The tonestack circuit of the JCM800.

This filter network design is shared by many other classic amps (Fender, Vox, Marshall, …). Only minor parameters differ from one amp to another: location of the tonestack in the amp circuit –after the first stage of the preamp on classic Fender amps- or the values of the different resistors and capacitors.

For our WebAudio amp designer, we explored different approaches: First, a transpiled version of the C++ Yeh's implementation[7] of the JCM800 tonestack to asm.js running in a WebAudio ScriptProcessor node. A minimum of 4ms in latency was introduced and we sometimes encountered buffer glitches when users operated the bass, medium and treble knobs from the amp GUI. Thus, as a simpler and alternative solution, we proposed to use a set of biquad filters in series: treble filter (high shelf, 6.5 kHz) goes into medium (peaking, 1.7kHz) into bass (low shelf 100Hz). The types of filters and their parameters are tuned to approximate the frequency response of the real tonestack (using the tonestack calculator tool[8] to compare). The result is not a completely faithful replication of the real tonestack circuit, but was quite well assessed by our testers[9] that found it easy to use and convenient to shape their sound rapidly. Finally, we also implemented the tonestack in an AudioWorklet node when the first implementations became available.

First improvement - the tonestack from Figure 8 runs now in WebAssembly inside an AudioWorklet and is fully customizable: we took advantage of the common plugin framework we developed - as described in Section 3 - to integrate an alternative family of tonestacks developed in FAUST[10] and transcompiled into AudioWorklet/WebAssembly. This approach allows to play with different tonestacks that are now quite faithful to their original versions based on the design presented in Fig. 8 (Marshall, Fender etc.). With this approach, one can also experiment easily with new topologies for the filter network.

Second improvement - the GUI allows to choose 'on the fly' among different tonestack implementations / settings: we can now select which tonestack to use: the "classic" emulation of the tonestacks from Figure 8 with a Fender, Marshall, Vox or other settings, or to use a different implementation such as the one we made earlier with filters in series.

Third improvement - tonestack and preamp can be swapped in the signal chain: namely, several real guitar amp designs locate the tonestack *before* the preamp, enabling 'bipolar' behaviors with "harsher/modern metal" distorted sounds when the preamp is cranked and "sparkling" cleaner sounds when the preamp is not saturated[11].

2.3 The Power Amp

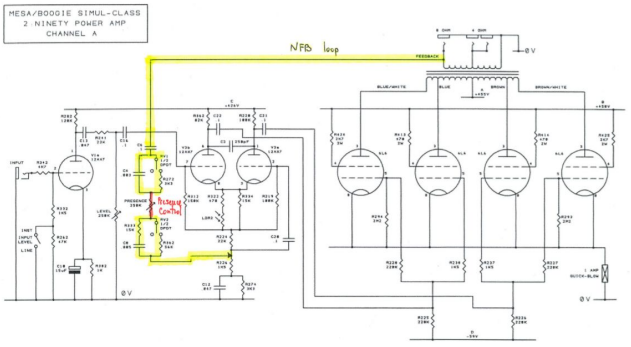

Figure 9. A Power Amp from the Mesa/Boogie Simul-Class 2:90. The Negative Feedback Loop (NFB) is highlighted.

Main improvement - the Power Amp stage is now fully simulated! In our previous implementations [1,2], the power amp stage was coarsely simplified and quite overlooked: we were just using a gain node (for the Master Volume) and a waveshaper node in series. The presence was simulated by an extra biquad filter node in the tonestack chain. We will explain now why the power amp is a critical stage to emulate and how we did it, despite the inconsistencies in WebAudio implementations we faced from one browser vendor to another.

The Power Amp stage, controlled by the Master Volume and Presence knobs, has a critical impact on the overall tone and dynamics, as the kind of sound you can get solely from the preamp stage remains on the lean side of distortion. Namely [5,8], *when you turn up a tube amp using the Master Volume*

[7] Available at: http://quitte.de/dsp/caps.html#ToneStack

[8] http://www.duncanamps.com/tsc/

[9] http://wasabi.i3s.unice.fr/AmpDesigner/focusGroup.html

[10] https://github.com/smoge/gxplugins

[11] Some video to see how hi-gain sounds can be obtained by swapping the preamp and tonestack: https://youtu.be/-NdMdJQx2Bw?t=4m11s see also https://youtu.be/PiOD7n3g-Qs

knob, the power tubes distort more and more and the tone gets thicker and less controlled. And as you get closer to power saturation (power tubes clipping and/or output transformer core saturation up to the famous 'grinding' effect, with its mix of saturation/oscillations), the output dynamics also get restricted, leading to 'muddy' distortions and 'heavier' sound.

Most guitar tube power amplifier stages (such as the Mesa/Boogie SC 2:90 shown in Fig. 9) rely on a phase splitter (here a long-tailed one) feeding a classical push-pull of pentodes or beam tetrodes (a set of four 6L6 tubes in ultra-linear topology, in the amp from Fig. 9). The power tubes are devices with high impedance output to be matched with the low impedance of the speakers which is usually done by using a push-pull transformer to limit core saturation. The 2nd harmonic (and all the even ones in general) are naturally cancelled by a well-balanced push-pull topology [5]. Besides, compared to triodes, the use of pentodes or beam tetrodes enhances the third harmonics (and all the odd ones) [4]. We end up with a spectrum mostly composed of odd harmonics that can be reasonably simulated using waveshapers with symmetrical transfer functions. However, if one wants to re-introduce even harmonics to get a warmer, more harmonious sound (as with the JCM800), one may instead use asymmetrical transfer functions, to get good simulations of the unorthodox but common use of slightly mismatched power tubes introducing even harmonics from a now slightly unbalanced push-pull.

Negative Feedback and Presence: in most amps, for the price of smaller total gain, a negative feedback loop (NFB) is added so as to tame the unwanted harmonics and to clean the output sound from unwanted distortion[12]. NFB is achieved by tapping a part of the output signal from the transformer secondary and by *feeding it back* with a 180 degree phase shift in the splitter stage (see the highlighted route in Fig. 9). Amps without NFB may be more lively (higher gain) with more progressive distortion (and less temporal smearing), but they also get more ragged and less tamed when pushed to saturation. The NFB loop smooths out the sound by reducing the level of distortion in the spectrum parts where it is most disturbing (typ. mid-range) [5].
Now, in guitar amps with NFB, *the presence control aims to restore brightness/life and sharpness to the power stage*. The behaviour of this control is quite different from the tonestack, which is only shaping the preamp output. Presence works as a 'holistic' brightness control via modification of the negative feedback loop. The taming effects of negative feedback is reduced above a certain frequency (determined by the RC network in the loop). With less negative feedback in the higher frequencies, the sound gets stronger, more crisp and dynamic. *Unlike a typical treble control which is subtractive (shunting the treble part of the signal to ground), the presence control is pseudo-additive in that it hinders some parts of the subtractive properties of the negative feedback loop*.

In our previous implementations of a JCM-800, the presence was coarsely simulated using a peaking filter at 3.9kHz in the tonestack. *Now, for our new guitar tube amp designer, we implemented a complete power amp simulation with bona-fide presence control and negative feedback loop. The presence filter is fully customizable and can be controlled on-the-fly with a graphical EQ (Figure 10) to choose where you want the presence increased. The total negative gain range in the NFB can also be adjusted using a slider.* These tools (NFB and Presence) are rather sensitive (larsens may arise easily, this is why we provide GUI controls to adjust/restrict them) but Presence control is also quite powerful and spectacular to shape the final sound.

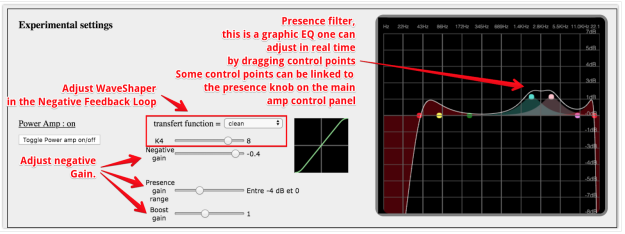

Figure 10. Power Amp NFB settings can be dynamically modified/adjusted on the fly with no glitch.

The WebAudio implementations of NFB and Presence were difficult due to some WebAudio API limitations and divergences/bugs in how browsers parse the WebAudio graphs. WebAudio API specs require any loop in the graph to have at least a delay node[13]. Accordingly, without delay node, Firefox stops rendering the graph, while Chrome does not complain but adds, behind the scene, a 3ms delay. Also, the signal is always processed with a minimum 128-frame audio buffer, which means that the minimum value for a delay is 128 / sampling rate, roughly 3 ms. For a proper implementation of the NFB, a 1-sample delay would have sufficed. Now, quite strangely, when adding a 3ms delay in the loop to conform to the specs, this brings slightly different coloring of the amps between FF and Chrome. As a perspective, we plan to re-implement the NFB loop as an AudioWorklet to allow a genuine 1-sample accurate processing (typ. by using FAUST).

2.4 The Reverb and The Speaker Cabinet

These parts were already well described and implemented in [2]. We have just added some new models for the sake of exhaustivity.

2.5 Evaluations and Perspectives

Similarly to what we did with our previous implementations [1,2], a set of professional guitarists compared the amps we designed, both the new and old implementations of our Power Amp with native commercial competitors. They all mentioned that the sound was even more realistic and was now much better controllable using the Master Volume and Presence knobs. They also insisted that the amp is more enjoyable to play with with better dynamic response. Opinions converged to say that this new version is an important improvement over the older ones. The full report of the user evaluations, also with videos, is available online[14].

Along with the new version of the NFB/Presence, new preamps and power amp models will also be added (typ. single ended versions like Vox AC4, Fender Champ or Gibson Gibsonette are on the radar screen). At the user interface level, we plan to add new features to allow customization of the end user GUI in the Amp designer, for example using graphic themes and templates.

3. PLUGINS: AMPS, AUDIO EFFECTS, INSTRUMENTS

Often, guitarists associate their guitar amps with audio effects brought by 'pedals' plugged between the electric guitar and the amplifier. Figure 11 shows the typical guitar rig that the late artist named Prince used in some of his live performances: nine

[12] http://www.aikenamps.com/index.php/designing-for-global-negative-feedback

[13] Step 4 from: https://webaudio.github.io/web-audio-api/#rendering-loop

[14] http://wasabi.i3s.unice.fr/AmpDesigner/focusGroup.html

FX pedals and a multi-channel amp[15]. These pedals are assembled on a so-called "pedal board" located on the floor, controllable by foot. In the native audio application world, these pedals and the amplifier would typically be "plugins", and the application that assembles them would be a "host", generally a Digital Audio Workstation (DAW) such as Ableton Live or Logic.

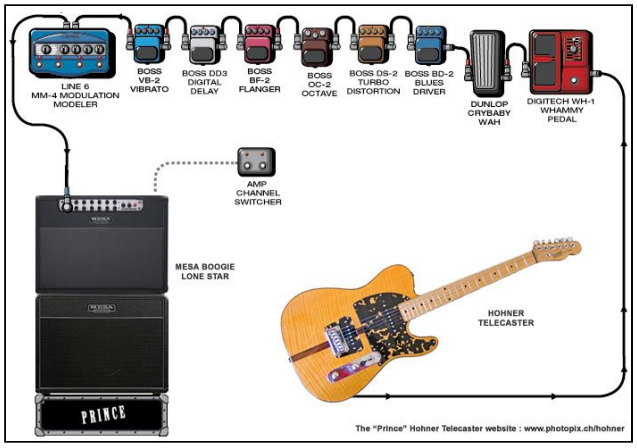

Figure 11. Prince's live guitar setup from Telecaster through effects pedals to the multi-channel amplifier.

Several plugin formats[16], both open and proprietary, exist today (VST, Apple AudioUnit, Avid's AAX, LV2, and meta standards such as JUCE or IPlug). All allow similar effects, however, usually in non-overlapping ways and are not quite suited as it is for Web applications. In 2003, the GMPI working group was started to gather the best parts of each existing standard and to bring them together into a cross-platform specification[17], later partially implemented by the LV2 (LADSPA version 2) open plugin standard (mainly adopted by the Linux community).

The WebAudio community would benefit from such a high-level "audio plugin" standard. Though we aim to introduce the functionality offered by the concept of native audio plugins to the Web, different environments require different approaches to the API design. A new API provides an opportunity to improve upon some aspects of native APIs, while taking into account the work done by the GMPI group and by the LV2 community, who tried to get the best of all other native standards.

This motivated us (and some other groups [11]) to collaborate with the researchers who created FAUST [10] and WebAudioModules [9] (WAMs) and other audio/WebAudio professional developers (such as g200kg[18]) to write a "proposal for an open WebAudio plugin standard" [3]. We also compared our proposal with the most recent/modern LV2 open standard and with the GMPI Final Draft Proposal[19].

This joint-work converged into a proposal that comes with a specification, a small SDK, an online plugin validator (Figure 12), a set of small examples that show how to discover plugins from repositories, how to instantiate a simple headless plugin, or a plugin with a GUI, and how to connect plugins together[20]. The main characteristic of WebAudio plugins is that they are identified by a unique URI and can be used locally or remotely by a host application without necessarily downloading and installing them on the host machine. For example, the online plugin unit tester from figure 12 can take a plugin URI that is located on a different server, all the code will come dynamically and if tests are passed, the plugin can be readily demoed in the page, and published without risks on a plugin repository.

Figure 12. Online plugin tester: enter the plugin URL to validate the plugin before publishing to a repository.

For this project, as a proof of concept, we developed from scratch a set of diverse pedal audio effects (in addition to the amp simulators described in section 1) and we also ported / adapted audio units (effects or virtual instruments) written by other developers using different programming languages.

Figure 13. OBXd Oberheim emulation, originally written in C++ and ported to WebAssembly by webaudiomodules.org.

Some plugins were written in JavaScript using WebAudio high level nodes (an overdrive pedal, a recreation of the quadrafuzz pedal[21], a stereo delay, a flanger, a chorus, some amp simulators presented in section 1, the Tiny Synth -a general midi synthesizer written by g200kg[22]), or compiled to WebAssembly / AudioWorklet nodes and originally written in in C/C++ using different native plugin SDK (four synthesisers from webaudiomodules.org such as the Oberheim OBXd emulation shown in figure 13, a Yamaha DX7, etc.), or written using the popular FAUST DSP language (Figure 13). We plan to develop scripts to port MAX DSP/Pure Data effects/instruments and to use some results from popular DSP environments such as CSound or Super Collider.

[15] http://makesenseofguitarpedals.pbworks.com/w/page/25461629/FrontPage

[16] http://www.steinberg.net/en/company/developers.html, http://www.juce.com/, https://github.com/olilarkin/wdl-ol, http://lv2plug.in/ns/

[17] http://retropaganda.info/archives/gmpi/

[18] https://www.g200kg.com/

[19] https://tinyurl.com/yd5fedrc

[20] https://wasabi.i3s.unice.fr/dev/

[21] A popular multi-band distortion described in "Do-It-Yourself Projects for Guitarists", by Craig Anderton, GPI BackBeat Books, 1995.

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

4. VIRTUAL PEDAL BOARD

Along with our "10-lines of code long" proofs of concept hosts, a "virtual pedal board" Web application was developed as a more ambitious host that gives the opportunity to create more complex sounds and configurations by interactively chaining the plugins (Figure 16). This pedal board host handles the discovery of plugins from repositories (local or distant), global sound card I/O and gain adjustments, plugins' life cycles and interconnexions, saving and restoring states/banks/presets.

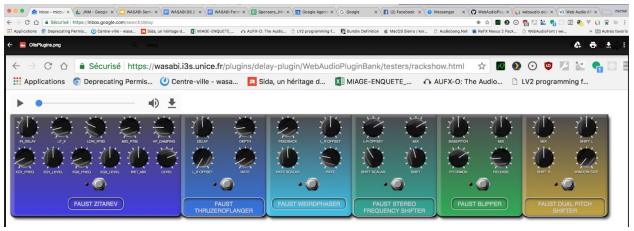

Figure 14. FAUST pedals, packaged as WebAudio plugins.

One can create instances of plugins by dragging and dropping their thumbnails into the main area. We can then position them, connect them together, adjust each plugin individually by using their GUI (knobs, sliders, switches), ... Now, by assembling an amplifier simulator, a reverb, a fuzz and a stereo delay, David Gilmour's sound on the solo of the song "Money" by Pink Floyd is within reach!

We did also contribute to the *webaudiocontrols library*[23] by adding MIDI support to all the GUI elements it offers (knobs, switches, etc.). Hence, the plugins received the possibility of having their GUI controlled remotely via any MIDI controller (Figure 17). Around the plugins, we also propose to improve the plugins interactions with MIDI synchronization inside the pedal board, to develop more MIDI based plugins (generators, midi sequencers, drumboxes...) and to introduce a way to publish and share online pedal board bank / patches.

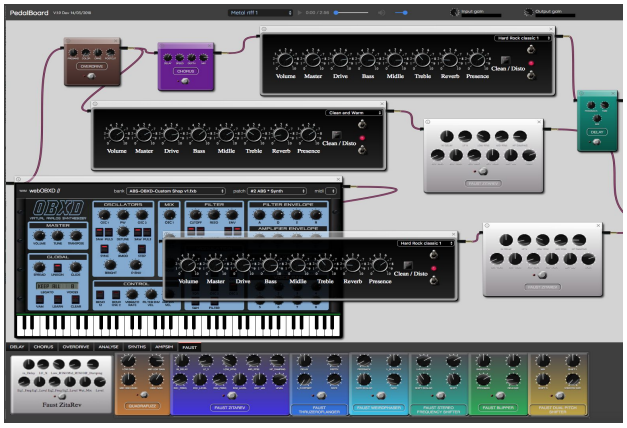

Figure 15. Plugins inside a virtual pedal board.
In black: our amp simulators, created by the Amp Designer

Figure 16. MIDI controllers have been tested for controlling WebAudio plugins' GUIs.

5 - CONCLUSION

We have presented here a virtual tube amp designer able to emulate each stage of a tube guitar amp, allowing multiple configurations inspired by existing hardware amps. The resulting simulations have been evaluated by professional guitar players and compared to some state of the art native counterparts. Furthermore, this amp designer is also a good pedagogical tool to illustrate and to explain to students or hobbyists how a guitar amplifier works. Each stage can be customized, bypassed or moved inside the signal chain, and the result can be heard instantly. Several models of preamps, tonestacks and power amps stages have been provided - more could be easily added - to further increase the range of sounds and dynamic responses achievable. In addition to this work, we converged to a proposal for a WebAudio plugin standard on which we have been working for several months, and illustrated the results with a host application: a "pedal board" which discovers plugins located on local or remote repositories, and allows the musician to assemble them into a "plugin graph" controllable with the mouse or with MIDI controllers. Some plugins in our demos are pure JavaScript / WebAudio, but we have also included plugins written in FAUST and also plugins that are WebAudioModules (VST/JUCE etc. ported to WebAssembly/AudioWorklet).

6. ACKNOWLEDGMENTS

French Research National Agency (ANR) and the WASABI project team (contract ANR-16-CE23-0017-01).

7. REFERENCES

[1] Buffa, M., Demetrio, M., and Azria, N. 2016. Guitar pedal board using WebAudio. In *Proc. 2nd Web Audio Conference* (WAC 2016), Atlanta, USA.

[2] Buffa, M. and Lebrun, J. 2017. Real time tube guitar amplifier simulation using WebAudio. In *Proc. 3rd Web Audio Conference* (WAC 2017), London, UK.

[3] Buffa, M., Lebrun, J., Kleimola, J., Larkin, O., and Letz, S. 2018. Towards an open Web Audio plug-in standard. In *Companion Proc. The Web Conference 2018* (WWW '18). Lyon, France. (April 23--27, 2018). 759-766. DOI= https://doi.org/10.1145/3184558.3188737

[4] Barbour, E. 1998. The cool sound of tubes [vacuum tube musical applications]. *IEEE Spectrum*, *35*(8), 24-35.

[5] Kuehnel, R. 2005. *Circuit Analysis of a Legendary Tube Amplifier: The Fender Bassman 5F6-A*. Pentode Press. 2005.

[6] Yeh, D.T. and Smith, J.O. 2006. Discretization of the '59 Fender Bassman tone stack. In *Proc. 9th Int. Conference on Digital Audio Effects* (DAFx-06), Montreal, Canada.

[7] Pakarinen, J. and Yeh, D.T. 2009. A review of digital techniques for modeling vacuum-tube guitar amplifiers. *Computer Music Journal.* 33(2), 85-100.

[23] https://github.com/g200kg/webaudio-controls, a WebAudio UI widget library.

[8] Denton, D. 2013. *Electronics for Guitarists*. Springer Science & Business Media. 2013.

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

[10] Letz, S., Orlarey, Y., and Fober, D. 2017. Compiling Faust Audio DSP Code to WebAssembly. In Proc. *3rd Web Audio Conference* (WAC 2017), London, UK.

[11] Jillings, N. & al. 2017. Intelligent audio plugin framework for the Web Audio API. In Proc. *3rd Web Audio Conference* (WAC 2017), London, UK.