Guitar pedal board using WebAudio

Michel Buffa, Maxime Demetrio, Nouriel Azria

Guitar pedal board using WebAudio

Michel Buffa
University of Nice
France
buffa@i3s.unice.fr

Maxime Demetrio
University of Nice
France
Maxime.demetrio@gmail.com

Nouriel Azria
University of Nice
France
azria.nouriel@gmail.com

ABSTRACT

The proposed demo is a guitar pedal board coded from scratch and inspired by the Guitar FX Chrome Application[1]. The demo is functional and proposes a graph based GUI for assembling effects such as: guitar amplifier emulation, Auto Wah, Delay, Distortion/Overdrive, Reverb, Chorus, etc. It includes a preset management system and saves incrementally, client side, any change in the parameters of the current preset. The latency is system and browser-dependent, but on a Mac Book Pro / Google Chrome the latency is as low as 14ms, making it playable in real time. The MVVC design is interesting for modularity and makes easy adding new effects. During the demo a guitar will be plugged into a computer using an external sound card, and the pedal board will process the sound in real-time. Lots of efforts have been put on the GUI and ease of use of the application, as well as on the core effect: the guitar amplifier simulation. The project is called "Guitar Processor" and is open source[2].

Introduction and State of the Art

Some applications showed that Web Audio could be used for processing real time input. Chris Wilson's "Input Effects" demo was one of the first to propose real time sound processing effects written with Web Audio, and proposed implementations of famous effects such as Delay, Distortion, Wah, etc. This webapp did not allow to chain effects but proved that low latency processing could be achieved. However, getting close to the sound of a real guitar amplifier is a real challenge that Chris Wilson's example did not address.

Many papers have been written about vacuum-tube guitar amplifiers modeling [1, 6], and about the particularities of linear and non-linear distortion effects suited for guitar [2, 3, 4, 5]. Some works such as James J. Clark "Advanced programming techniques for modular synthesizers" book, are not focused on guitar but cover in deep the different approaches for achieving a distortion effect on a signal.

Figure 1: a typical, single channel, guitar amplifier.

Wikipedia gives a rather good description of the high-level design of a guitar amplifier: *"Typically, guitar amplifiers have two amplifying circuit stages and in addition frequently have tone-shaping electric circuits, which usually include at least bass and treble controls, which function similarly to the equivalent controls on a home hi-fi. More expensive amplifiers typically have more controls for other frequency ranges, such as one or two "midrange" controls and a "presence" control for very high frequencies. Some guitar amplifiers have a graphic equalizer, which uses vertical fader controls, which can control many frequency bands. The first amplifier stage is a preamplifier stage (there may be more than one), which amplifies the guitar signal to a level that can drive the power stage. The power amplifier or output stage produces a high current signal to drive a loudspeaker to produce sound that the guitarist and audience can hear."*

When we started the project of writing a guitar pedal board, we began with the amplifier simulation, that is the "master effect". We discovered that many authors of native guitar amplifier plugins (VSTs, etc.), including famous makers such as IK Multimedia (creators of the Amplitube guitar amp plugin), or Lepou (author of many guitar amp simulators) chose to use the "analog simulation" approach, as stated by an engineer[3]: "*The way we do it at IK Multimedia is a bit different. We do not rely on just schematics. Our engineers acquire every piece of gear they model, and take precise measurements of every component and every interaction. Then they custom build dynamic, non-linear algorithms to replicate every nuance of the behaviors. And finally, with the help of our co-branding partners such as Fender,*

---

[1] Guitar FX Chrome Application : https://chrome.google.com/webstore/detail/guitarfx/nbjaofogegcpkeobcnpdkkpdioogeajj

[2] Guitar Processor : https://bitbucket.org/Felours/ntdp-projet_stage_guitarfx

[3] Forum "guitar amp modeling": http://www.guitarampmodeling.com/viewtopic.php?f=51&t=7006&start=125

Orange, Soldano, and Ampeg, to name a few, we fine tune everything by ear until our models sound identical to the original hardware at the same settings."

For this project we did not have time nor ambition to go to such a level of accuracy, and we used an approach based on Wave Shapers[4]. This approach changes an audio signal by mapping an input signal to the output signal by applying a fixed or variable mathematical function, called the *shaping function* or *transfer function*. The Web Audio specification proposes such a function that has been copied and pasted in most of the examples published that use the WaveShaper node. The problem is that the sound obtained is not rich enough for a demanding guitarist. Further work is needed. As stated by Lepou, who wrote many guitar amp plugins: "*Of course an amp shapes the incoming wave and can thus be seen as a waveshaper. However, the waveshaping function can not be applied to all the frequencies and intensities of the incoming signal…tubes cannot be treated as simple waveshapers. I think the important word is SIMPLE. In my opinion, a tube is a waveshaper but when put in a complete circuit the shaping function changes with frequency and intensity of the incoming signal. That's why a tube gain stage cannot be considered as a SIMPLE waveshaper* ».

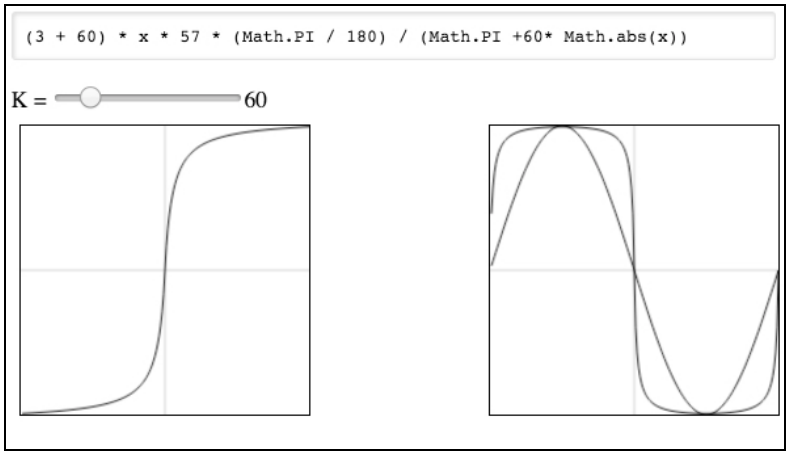


Figure 2: transfer function on the left, and on the right, the source (sinusoidal) signal, and the transformed signal.

In his book, James J. Clark describes the *quadrafuzz* circuit[5] that addresses in a simple way the fact that we need to shape differently the input signal depending on frequencies: we can split the signal into separate paths using simple filters (a low pass filter, two band pass filters and a high pass filter). We reproduced this part of the preamp using a set of Web Audio BiquadFilter nodes. The dynamic part of the behavior of the vacuum tubes has been simulated using an analyzer node for guessing the intensity of the signal that in turn is used for adjusting the amount of distortion in the waveshaper nodes.

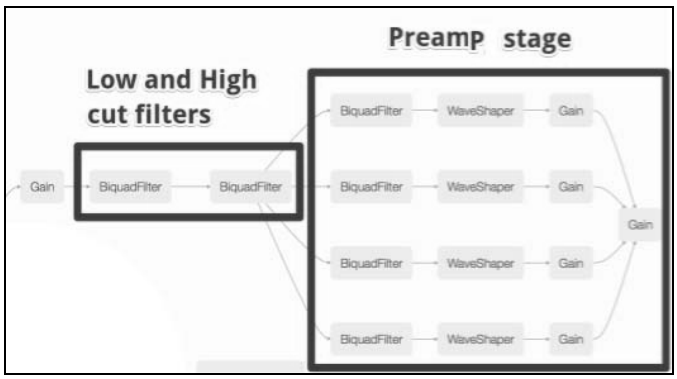


Figure 3: pre-amplification stage, notice the filters at the beginning for cutting too low and too high frequencies. The signal is then split into 4 frequency ranges, then shaped. The WaveShaper distortion parameters are adjusted depending on the intensity of the signal (detected by an analyzer node not presented here).

The signal is pre-filtered using a high and a low pass filter in order to cut the frequency at the extremities of the spectrum (< 20Hz and > 10.000 Khz).

The four biquad filters that split the signal have been pre-adjusted to some frequency ranges ([lowpass 70-300Hz], [bandpass 300-1200Hz], [bandpass 1200-3500Hz], and ([highpass 3500-7000Hz]), but can be tweaked, as well as the Q parameter of the filters, in order to "sculpt" the sound. Usually, such settings are not available to the end users of a real guitar amplifier, they are part of the DNA of the sound of some music brands such as Fender, Marshall, etc. (other parameters will play a role of course, speakers, for example).

---

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

[5] The quadrafuzz is a famous distorsion VST plugin edited by Steinberg, based on some real circuiry from the 70's.

The gain at the end of the preamp stage can be seen as the "master volume" of a real amplifier. In order to get clean sounds, our experience showed that we need to have 1) low values for the distortion parameter of the WaveShaper nodes, as well as 2) a very low master volume (< 0.1). In order to get a distorted sound, raise the master volume and lower the main volume. Like with a real amplifier.

The main volume is a simple gain node located right before the speaker simulator. We also inserted a reverb in our amplifier simulation, before the main volume (where the effect insertion loop is usually located, in real amplifiers). Reverb and speaker simulations are achieved using a convolver node and impulse files.

Figure 4 shows the GUI of the first prototype (with many parameters not suited for an end user, but usueful for understanding how amplifiers' tones are created).

Latency and Web Audio problems

The getUserMedia API enables to use any sound input[6]. However by default, some browsers, such as Google Chrome implement an automatic gain adjustment algorithm[7] that is not suited for music applications. This option can be switched off using a constraint passed to the `navigator.getUserMedia` function (set the `echoCancellation` property to false). However, when we used this constraint, in the current implementations, it has not been possible to switch inputs (we reported the bug).

The second problem is latency. The intrinsic latency of a system (browser / OS / Sound Card) can be measured using Stephen Band's sound.io/latency[8] Web application, that measures the output-to-input latency (making a sound and listening for it), and results vary deeply from one configuration to another. Google Chrome/Mac OS x gives 9ms, where Firefox on the same computer gives 140ms. A "good latency" for playing real time an instrument is noticeable above 10ms. Add to this all the processing in our amp simulation and effects. So we did our best to avoid using the Web Audio nodes that involve additional latency. Unfortunately, this information is not specified, nor the nodes expose in a way or another their latency. Some APIs like JUCE propose some methods with its components[9] for "guessing the latency". In an interesting thread on stack overflow[10], it appears that only specific Web Audio nodes could involve latency overheads. They are: the WaveShaper node when set in oversampling mode (avoid oversampling!), as it has to look ahead a few samples to make some mean computations, the Delay (of course!), and the convolver node with large impulses (use small impulses!).

In our guitar amp simulator, we followed these good practices and manage to get a very low latency processing time, nearly unnoticeable, on Mac OS/Google Chrome. We measured the latency (make a sound and listen to it) and we got 15ms of latency (30ms round trip).

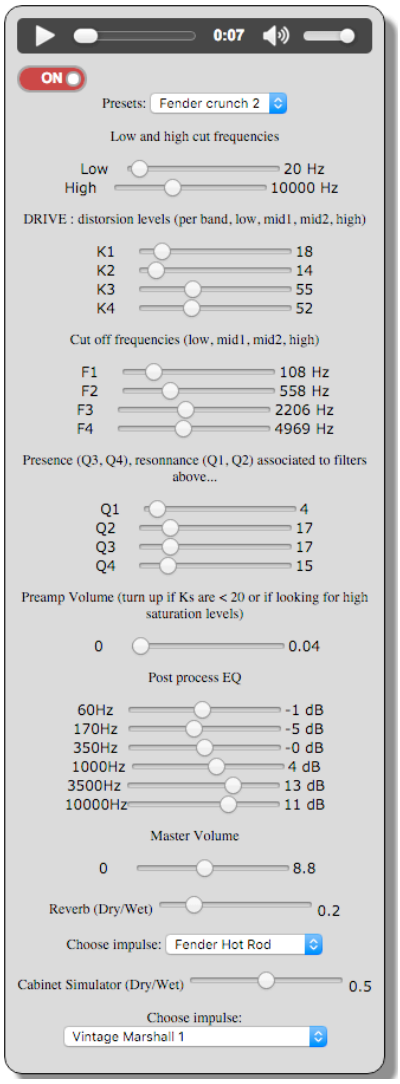


Figure 4: the first prototype of the guitar amp simulator

Pedal board application

Finally, in parallel to the work conducted in order to play with the "core sound" of a guitar amplifier, fixing latency and sound quality problems, we developed a graph based GUI for designing a chain of effects. Most of the effects are based on examples found on the Web; small improvements have been done to suit our ears.

This application uses the jsPlumb JavaScript library for managing graphically the graph, for connecting nodes, etc. When a user clicks on an effect, a set of controllers (knobs, switches, preset menus) appears, enabling fine-tuning of the effects. The current state of the effects is saved incrementally in IndexedDB each time a user interacts with the application. In addition we developed a complete preset manager interface. The GUI has been inspired by the Guitar FX Chrome Application (which is not open source). The guitar amplifier presented previously comes this time with a simplified GUI with "classic" knobs and EQ, making it easier to use.

---

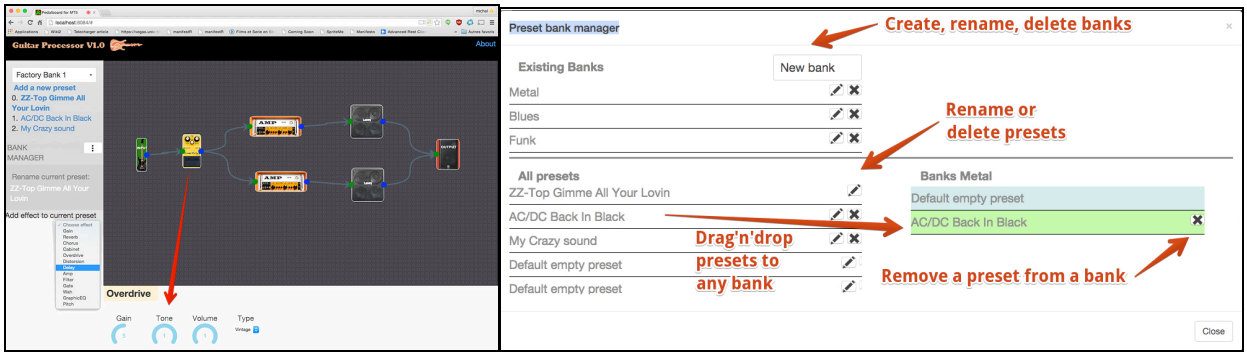
[6] https://webrtc.github.io/samples/src/content/devices/multi/

[7] http://googletesting.blogspot.fr/2015/10/audio-testing-automatic-gain-control.html

[8] sound.io/latency

[9] getOutputLatencyInSamples from JUCE: http://www.juce.com/api/classAudioIODevice.html#a693804fbf5a7cceb31ece10a9f03bd11

[10] http://stackoverflow.com/questions/21428511/playing-and-recording-with-a-constant-latency-on-browser-with-javascript-usermed



References

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

[2] Holmes, B., & van Walstijn, M. Improving the robustness of the iterative solver in state-space modeling of guitar distortion circuitry. Proc. of the 18th Int. Conference on Digital Audio Effects (DAFx-15), Trondheim, Norway, Nov 30 - Dec 3, 2015.

[3] Cheng-Hao C. A Guitar Overdrive/Distortion Effect of Digital Signal Processing. http://ses.library.usyd.edu.au/bitstream/2123/7624/2/DESC9115_DAS_Assign02_310106370.pdf

[4] Macak, J., & Schimmel, J. (2010, September). Real-time guitar tube amplifier simulation using an approximation of differential equations. In Proceedings of the 13th International Conference on Digital Audio Effects (DAFx'10).

[5] Yeh, D. T., Abel, J. S., Vladimirescu, A., & Smith, J. O. (2008). Numerical methods for simulation of guitar distortion circuits. Computer Music Journal, 32(2), 23-42.

[6] Cohen, I., & Helie, T. (2010, September). Real-time simulation of a guitar power amplifier. In 13th Int. Conference on Digital Audio Effects (DAFx-10).

[7] Hotz, M. A Study of Tube Amplifier Modeling Using Nonlinear Wave Digital Filters. Master student thesis. https://spsc.tu-graz.ac.at/sites/default/files/tube_amp_modeling.pdf

[8] Clark, J. J. (2003). Advanced programming techniques for modular synthesizers. Internet, [Besökt Maj 2005], URL: http://www.cim.mcgill.ca/clark/nordmodularbook/nmbooktoc.html.