Real time tube guitar amplifier simulation using WebAudio

Michel Buffa, Jerome Lebrun

▶ **To cite this version:**

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⟩

HAL Id: hal-01589229

https://univ-cotedazur.hal.science/hal-01589229v1

Submitted on 18 Sep 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

Real time tube guitar amplifier simulation using WebAudio

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

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

ABSTRACT

This paper presents a tube guitar amplifier simulation made with the WebAudio API, that reproduces the main parts of the Marshall JCM 800 amplifier schematics. Each stage of the real amp has been recreated (preamp, tone stack, reverb, power amp and speaker simulation, and we added an extra multiband EQ). The "classic rock" amp simulation we built has been used in real gigs and can be compared with some native amp simulation both in terms of latency, sound quality, dynamics and comfort of the guitar play. Unfortunately, as of today, low latency can be achieved only with certain configurations, due to audio driver limitations of current browsers on certain operating systems. The paper discusses the latency problems encountered with WebAudio, common traps, current limitations, and proposes some solutions.

The final web based simulation has been compared with native re-creations of the same amp model (including commercial products such as GuitarRig, the JCM800 amp included in GarageBand or the open source Guitarix amp sim that runs on Linux), and with a real amp: the Yamaha THR10 that comes with a model of a Marshall amp. We conducted both quantitative evaluations (measure of the "guitar-to-speaker" latency, group delay, frequency response analysis) and qualitative evaluations with real guitar players who compared, guitar in hands, the different simulations in terms of sound quality and dynamics, and more generally "how they feel playing guitar with these simulations". The amp is open source[1] and can be tested online[2], even without a guitar (it comes with an audio player, dry guitar samples and a wave generator that can be used at input). The Web page contains links to the source code repository, tutorial videos and a complete report of the measures we made, with different configurations (various soundcard, operating system, browsers), that is summarized in this paper. Figure 1 shows the current GUI (with optional frequency analyzers and oscilloscopes we used to probe the signal at different stages of the simulation).

Our initial goal was to evaluate the limits of the WebAudio API and see if it was possible to design a web based guitar amp simulator that could compete with native simulations.

1. INTRODUCTION

Guitar amplifier digital models became popular with devices such as the pod series by Line6 in the early 2000s, or more recently with the Axe FX-II amp modeler by Fractal Audio Systems: an all-in-one preamp/effects digital processor that contains a vast virtual inventory of hundreds of vintage and modern guitar amps. In 2002, Amplitube, an audio plugin commercialized by IK multimedia, was

Figure 1: web based tube amp simulation, GUI with visualization tools.

[1] https://github.com/micbuffa/WebAudio-Guitar-Amplifier-Simulator-3

[2] https://wasabi.i3s.unice.fr/AmpSim3/and a version with measure tools activated: https://mainline.i3s.unice.fr/AmpSimFA

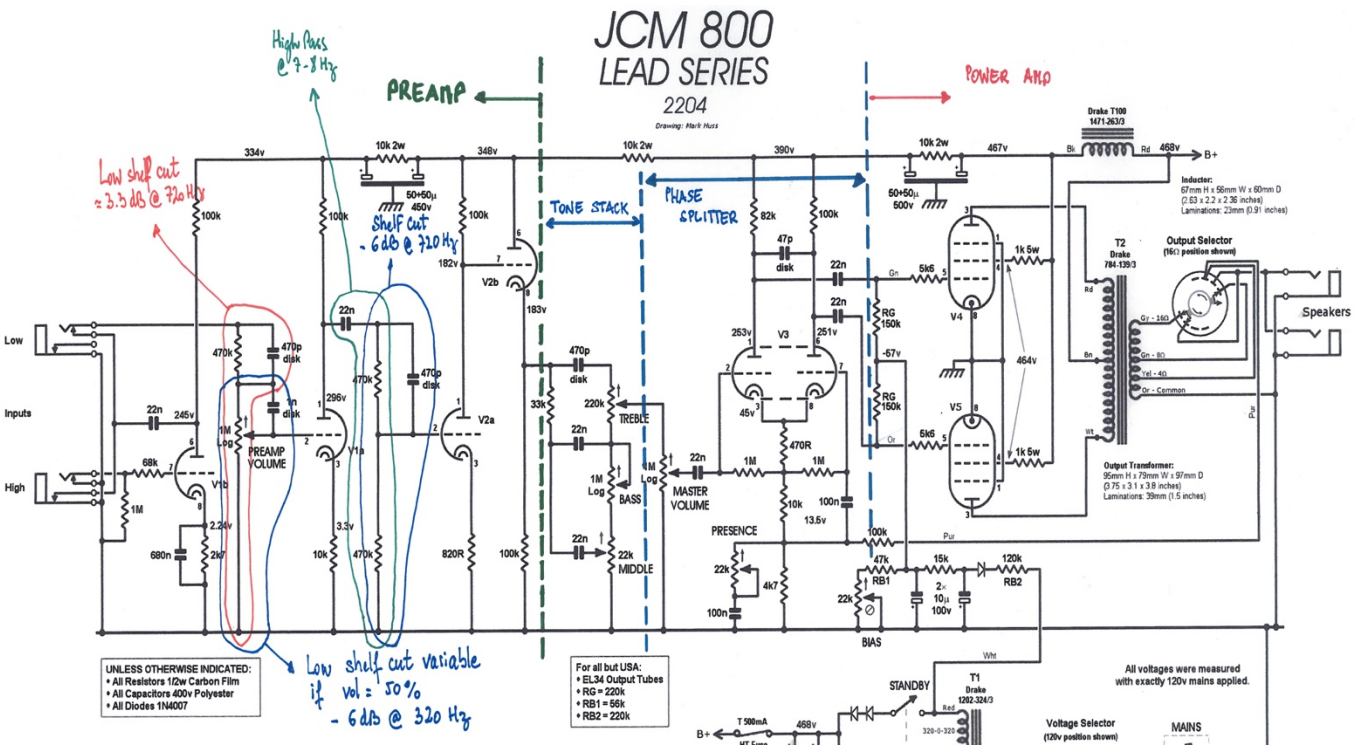

Figure 2: Marshall JCM800 schematics.

From left to right we distinguish the different stages: preamp, tone stack, power amp

the first popular 100% software amp simulation on the market, followed soon by Guitar Rig by Native Instruments. Today we can find hundreds of native plugins (commercial or freeware, only a few are open source) for digital audio workstations, that simulates existing guitar amps, or are based on an original design by their authors.

Software amp simulations are popular in nomad situations, or when the production budget for recording is low, as they are cheaper and more flexible than their digital hardware equivalents, although purists claims that they'll never be the same as the real thing.

2. State of the art

In 2012, Google Chrome proposed for the first time low-latency access to live audio from a microphone or other audio input on Mac OSX, followed by a Windows implementation (with a bigger latency). Soon Opera, Firefox and more recently Microsoft Edge also implemented this features that relies now on the Media Capture and Streams API from W3C[3].

Chris Wilson's "Input Effects" demo[4] was one of the first to show real time sound processing effects written with WebAudio, and proposed implementations of famous effects such as Delay, Distortion, Wah, etc. This demo 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 examples 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 [9].

The common approach consists in modeling the different parts of a 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 (this "tone shaping" module is called the "tone stack"). 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."*

There are two main approaches for simulating the different parts of a guitar amplifier: one is called the technique of virtual analog (or physical modeling) and consists in entering the electronic schema in a tool like the industry standard SPICE analog circuit simulator, then translate the circuit into equations to be solved. These general equations are typically nonlinear differential algebraic equations

[3] https://www.w3.org/TR/mediacapture-streams/

[4] https://webaudiodemos.appspot.com/input/index.html

that may be solved using integration methods, roots solver algorithms, and sparse matrix techniques.

SPICE can produce C++ code ready to be executed. However, it is often necessary to make simplifications and optimizations to obtain a solution allowing real time processing. This is particularly the case with the modeling of vacuum-tubes used in guitar amplifiers and their interactions with other parts of the circuitry (see [1] and [17] for a review of common techniques).

Another technique consists in a higher-level emulation, in which "logical" parts are identified (filters, tubes, etc.) and emulated manually using separate models. This is in theory less accurate as some effects and interactions such as the current feedback effect of overloaded tubes or the action of the speaker impedance on the power amp/sound tone may not be considered. However, this approach is simpler and more adapted to WebAudio and its today limitations (custom processing on audio samples with the Script Processor node is not usable without introducing latency or glitches, for example). Furthermore, WebAudio proposes some high-level nodes (such as the Wave Shaper node and the biquad filter node) that can be used for modeling tubes and filters, and it has been shown that properly used, wave shaping techniques associated with oversampling and appropriate filtering, can give good results [18]. The famous pod XT effect processor by Line6 uses such techniques [1].

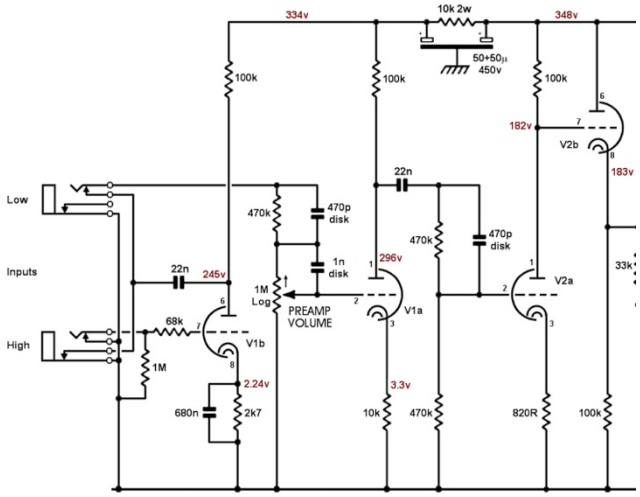

Figure 3: JCM800 preamp

As far as we know there is no previous work that tried to simulate a complete guitar amp using WebAudio. Pedals.io[5] is a JavaScript recreation of some classic audio effect pedals for guitarists (delay, chorus, overdrive, etc.), and we find nearly the same implementations of these effects in many WebAudio JavaScript libraries such as toneJS[6], tunaJS[7], pizzicatoJS[8], etc. The most advanced work we found is a Google Chrome application named GuitarFX[9] that proposes simple amp models altogether with a set of audio effect pedals, but does not recreate in detail each stages of a real amp (only one wave shaper per amp, for example).

3. Modeling the different amp stages

3.1 Preamp

The preamp of the Marshall JCM800 is shown on Figure 3. It is composed of several filters and two dual triodes (V1 and V2, typ. two 12AX7). The second, named v2a and v2b, located at the end of this stage (right of the figure), is a DC coupled cathode follower buffer, limiting clipping and acting as a linear driver of the tone stack (section 3.2). The most interesting part is made of the filters and the first dual triode v1a and v1b. From the left of the signal chain to the right, we have a low shelf filter with a -3.3dB gain at 720Hz, then another low shelf at -6dB 320Hz, and the first triode stage named v1b, followed by a high pass filter at 6-7Hz. This is the first part of the preamp. The low shelf filters cut the annoying frequency generated by the guitar, and the V1 part generates odd and even harmonics (v1a is mainly a gain, v2a amplifies and introduces harmonics). Even harmonics are important to perceive the sound as "warm, bluesy", and odd harmonics yield a more "harsh, gritty" sound. The mix of both even and odd harmonics is known to be the secret of the "warm punchy sound" you can find in "classic rock" and blues, if nicely distributed. To model the two parts of this first tube, we used a WebAudio wave shaper node with the asymmetric transfer function described in Pakarinen-Yeh's article [1] (with origins from a patent held by Doidec et al. 1998), shown in the left of Figure 5.

$$f(x) = \begin{cases} 0.630035, & 0.320018 \le x \le 1 \\ -6.153x^2 + 3.9375x, & -0.08905 \le x < 0.320018 \\ -\frac{3}{4}\left\{1-\left[1-(|x|-0.032847)\right]^{12}+\frac{1}{3}(|x|-0.032847)\right\}+0.01, & -1 \le x \le 0.08905 \\ -0.9818 & x < -1 \end{cases}$$

Figure 4: Asymmetric distortion function (Pakarinen, Yeh)

This function clips differently the negative and positive portions of the wave ("harder clipping" on the negative portion, while the positive portion of the wave is softly clipped). This asymmetry somewhat approximates the duty-cycle modulation seen in overdriven tube amplifiers. This asymmetry in clipping adds both even and odd harmonics, resulting in a richer tone that characterizes vintage tube amps. The output signal is no more centered as the asymmetry adds to the DC value. The high pass filter at 6-7Hz rectifies this signal and removes the hum noise that could have been amplified.

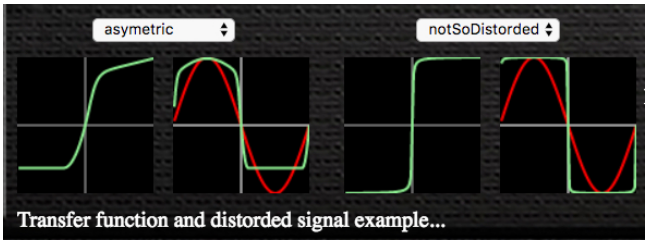

Figure 5: Transfer functions used for the two first tubes in the JCM800 preamp (v1a and v1b)

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** owner/author(s).

Web Audio Conference WAC-2017, August 21–23, 2017, London, UK.

© 2017 Copyright held by the owner/author(s).

[5] https://pedals.io/

[6] https://github.com/Tonejs/Tone.js/

[7] https://github.com/Theodeus/tuna

[8] https://alemangui.github.io/pizzicato/

[9] https://tinyurl.com/ljdhuqh

The second part of the preamp is made of another low shelf filter at 720Hz with a gain of -6dB. This time, for the second triode stage V2, we used a symmetric transfer function (in the example made with a *tanh* function) for generating more even harmonics.

In our implementation, we included 15 different transfer functions that can be set to the different preamp tubes (V1 or V2), giving more versatility to the original JCM800 preamp design. The ones shown in Figure 5 produce a sound rich in both even and odd harmonics, with a high level of distortion/overdrive, that can be heard in videos with guitar players who tested the simulated amp[10].

Filters have been implemented using standard WebAudio biquad filters.

Some care must be taken: during clipping, these transfer functions may produce harmonics which exceed half of the sample rate and wrap back around to the high frequencies, causing aliasing artifacts in the signal [1]. Some methods such as oversampling are known to allow a larger frequency range before aliasing occurs to allow higher harmonic components in the digital frequency domain. The WebAudio wave shaper node has a property named "oversampling", that unfortunately increases the latency (depending on its value "2x" or "4x"). Another approach for limiting aliasing is known as multiband distortion and consists in splitting the signal into several separate frequency bands and to apply different amounts or types of distortion in each subband [9]. We implemented this approach in a previous version of the WebAudio amp simulation [19], that indeed produced very nice clean sounds. This version based on an original design, not inspired by any existing amp, can be tried online[11]. The preamp we implemented for the work presented in this paper integrates optional oversampling, and we also measured that, in addition, our speaker simulation stage cuts most of the high frequencies that may result from aliasing. These measures have been confirmed during qualitative evaluations, as none of our human testers could hear a difference, with or without oversampling activated.

3.2 Tone stack

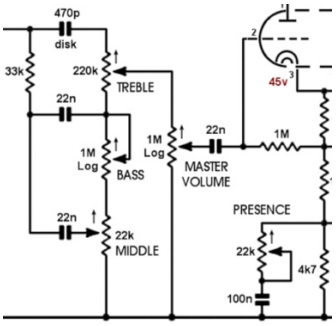

Figure 6: the tone stack circuit of the JCM800

In a reference paper about the modeling of the Fender Bassman amp tone stack[6], reproduced in the JCM800, David Yeh and Julius Smith explains that the circuit (bass, medium, treble) that shapes the sound of the electric guitar, presented in Figure 6, cannot be simulated accurately by filters in series or in parallel. Besides its simplicity, this circuit results in a complicated filter with each individual part influencing the other. Yeh and Julius analyzed the schematics symbolically using the technique of virtual analog, and proposed an exact solution: a filter that responds to user controls in the same way as the analog prototype.

As many classic Amps by Fender, Vox or Marshall share this design, we can find exact implementations of many different tone stacks (only the location on the amp circuit–after the first stage of the preamp on classic Fender amps-, or the values of the different resistors and capacities, changes). In our WebAudio amp, we transpiled the original C++ implementation of Yeh's code[12] from C++ to JavaScript, using emscripten[14], and used a WebAudio ScriptProcessor node to reproduce the exact tone stack of the JCM800. However, experimentations showed that a minimum of 4ms in latency was introduced, therefore we decided to explore further alternatives. In addition, we encountered sometimes buffer glitches while users operated the bass, medium, treble and presence knobs from the amp GUI (as the ScriptProcessor node runs in the GUI thread), this led us to remove temporarily this implementation (waiting for the upcoming AudioWorklet node that should give better performances and no glitches). We adopted as an alternative solution 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) into presence (peaking, 3.9KHz), we adjusted the types of filters and their parameters so that they approached the frequency response of the real tone stack (we used the tone stack calculator tool[13] for comparing). The result is not a faithful replication of the real tone stack circuit, but "does the job", our testers found it easy to use and managed to shape their sound rapidly. In a real JCM800 the presence filter is in the feedback loop between the input and output of the power amp stage, however we kept it in the series of filters when we switched the tone stack implementation.

3.3 Reverb

We used a convolver node with free reverb impulses[14]. We implemented this using a classic wet/dry audio graph to make the "room effect" adjustable. Several impulses are included in the online demo (Marshall JCM800 plate reverb, Fender spring reverb, etc.)

3.4 Power Amp

The power amplifier stage of the JCM800 (right of Figure 2) relies on a long-tailed phase splitter feeding a classical push-pull of pentodes (two EL34 or 6550). The 2nd harmonic (and all the even ones in general) are naturally cancelled by a well-balanced push-pull [11]. Also, compared to triodes, the use of pentodes enhances the third harmonics (and all the odd ones) [12]. We end up with a spectrum mostly composed of odd harmonics that could be reasonably simulated using symmetrical transfer functions [10]. However, if one wants to re-introduce even harmonics to get a better harmonious version of the JCM800, one should use instead asymmetrical transfer functions, which provides a good simulation of the classical use of slightly mismatched pentodes to get even harmonics from an unbalanced push-pull. We added a low gain (< 1) before our power amp stage to reproduce the "light" overdrive expected effect. As for the power amp part in real analog amps, in addition to the overdrive, an influence on the tone is linked to the impedance of the speakers in the cabinet. This impedance is due to the load of the power lamps and therefore "shapes" the sound. This document[16], written by the creator of Randall tube amplifiers

[10] https://wasabi.i3s.unice.fr/AmpSim3/userEvaluation.html

[11] Multiband distortion preamp in a WebAudio amp sim: https://mainline.i3s.unice.fr/AmpSim

[12] Available at: http://quitte.de/dsp/caps.html - ToneStack

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

[14] https://plus.google.com/+YoannPichard/posts/MTDzQgpSS9L

gives a general idea of the impedance of a speaker and how to model it. We did not implement this part yet as the speaker simulation has a much greater effect on the overall tone. While negligible, we plan to add it to our model as a future improvement.

3.5 Speaker simulation

We used a convolver node with cabinet impulses from the Redwirez commercial set[15]. Included in the online demo of the WebAudio amp is the matched JCM800 cabinet as well as some Fender and Vox cabinet impulses. We implemented this using a classic wet/dry audio graph to make the effect of the cabinet simulation on the final output signal adjustable. The quality of impulses affect the overall sound tone and lowers aliasing.

4. Evaluations

4.1 Latency

Paul Adenot from Mozilla gave a tutorial at the WebAudio Conference 2016 that covered all performance facets of the WebAudio API, and includes a whole section about latency [15]. Latency depends on the sound card / driver / size of the audio buffer / USB ports, on the Operating System / audio driver supported by the browser (low latency processing is not possible on Windows due to the WASABI audio driver being the only one available today on all browsers -no ASIO on windows, Firefox supports Jack audio on Linux, while not enabled by default, etc.), on the browser implementation (for example, the multi process design of Google Chrome adds latency to do inter process communication). Latency is also introduced by some WebAudio nodes: the delay node (obviously), the compressor node adds a fixed look-ahead of 6ms, biquad filter nodes add a two-frame latency that is negligible, wave shaper nodes, when oversampling is enabled, add latency, the panner node adds latency depending on some azimuth and elevation parameters, finally, the convolver node adds latency that depends on the size of the impulses.

In our implementation, we tried to keep latency as low as possible: and did not use any feature/nodes that could add latency except the convolver nodes. However, we used very small reverb (300-700k bytes) and speaker (10-20k bytes) impulses, that added no more than 1-2ms latency on our reference desktop.

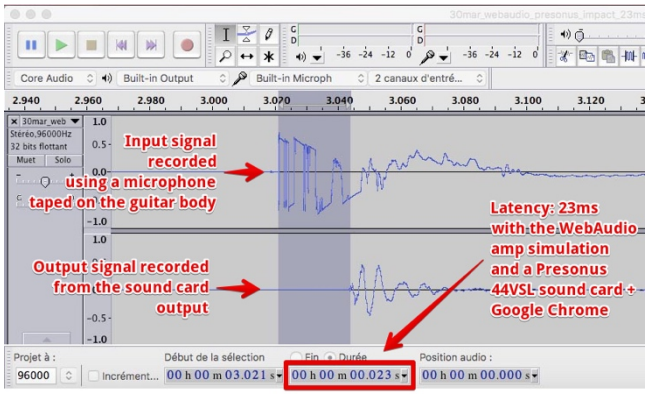

Figure 7: Example of measure in Audacity, here the WebAudio amp sim with Google Chrome and a Presonus 44VSL sound card.

We tested latency with a Mac Book Pro 2016, Mac OSX Sierra, 16GB ram, both with a Presonus 44VSL sound card, and an Apogee Jam sound card. We ran the WebAudio amp on a "clean" browser (no other tabs and processes running) and tried both with Chrome and Firefox Nightly. As of today, there is no means in the WebAudio API for setting the size of the audio buffer, so we used the default buffer size of 256 samples double buffered, that corresponds to 11ms of intrinsic latency (confirmed by the `baseLatency` property of the WebAudio context's value). We also measured the latency of Guitar Rig's JCM800 amp sim (as a plugin in GarageBand), and with the Clean Brit amp plugin that comes bundled with GarageBand (also a JCM800 replica). Buffer size is locked to 512 samples with GarageBand versions >= 10, the same value used by WebAudio in our test configurations. In addition, we compared with a Mod Duo Linux-based pedal and a Yamaha THR10 guitar amp as a reference. We measured the guitar to speaker latency, using a Panasonic WM61A electret microphone taped on the guitar body, next to the high E string (Figure 8), and a jack plugged at the output from the sound cards. We joined them using a Y wire plugged into a Sony PCM-M10 digital recorder at 96kHz/24bit

Figure 8: Latency measurement setup

We then hit several times the guitar body with a metallic rod, and recorded the result while going through different sound card and amp simulations (with settings that did not alter too much the signal). Figure 7 shows an example of measure in audacity after we loaded a wav file we just recorded. The selection gives a rather good approximation of the latency. However, we confirmed these measures using a Matlab script of our own that analyses the different Dirac shaped impulses to compute max/min and mean value of the latency as well as estimations of the group delay. Measures and .wav files are available online[16].

A summary of our latency measures is shown in Table 1. We compared latencies with an audio buffer of 256 samples double buffered (eq. 512) both with WebAudio and with native plugins in GarageBand (they both do not allow to change this setting). We also compared with two other configurations that allowed a much lower latency/buffer size. It is interesting to notice that even with the slightly higher numbers in WebAudio/GarageBand setups, real guitar players did not notice the latency with Google Chrome nor found it problematic for playing comfortably (this was not the case with Firefox Nightly that had a higher latency than Google Chrome[17], see next section). The digital plugin community usually estimates that you can feel the latency when it's higher than 10ms,

[15] http://www.redwirez.com/

[16] https://wasabi.i3s.unice.fr/AmpSim3/latency.html

[17] We used Google Chrome version 59.0.3053.3 dev (64-bit) and Firefox Nightly version 55.0a1 (2017-04-01) (64-bit) for Mac OS

and experimented guitarists could even feel it above 5ms. Presonus on the other side states[11] that "*our roundtrip latency of 9.7ms is still below the realm of human perception, and it shouldn't affect your performance*". See the Audio Anecdotes book [20] for a demystification of the perceived audio latency.

With the Yamaha THR10, a hardware guitar amp with amp modeling using a dedicated DSP, we measured a latency < 1ms.

The Mod Duo pedal[18] uses a Linux operating system with LV2 plugins to define virtual pedal boards. This hardware runs on a dual core ARM A7 1.0GHz, a dedicated Cirrus Logic sound card and uses a fixed audio buffer of 128 samples at 48Khz. Latency with no plugins, given by the constructor, is 9ms. We used it with the LV2 Guitarix JCM800 native plugin and a matched cabinet simulator.

Table 1. Measured guitar-to-speaker latencies

	Guitar Rig JCM800	Garage Band JCM 800	Guitarix JCM800	Web Audio Amp (Chrome/FF Nightly 03/2017)
Presonus 44VSL	17ms	19ms		23/32ms
Apogee Jam	19ms	26ms		26/35ms
ModDuo Pedal (Linux)			21ms	

4.2 Frequency responses and waveforms

We used sinusoid waves at different frequencies and amplitudes as an input signal, and plotted the frequency responses and waveform at different stages of the WebAudio amp. We then compared with outputs from a real JCM800 amp and outputs from the Guitar Rig JCM800, with similar settings. Frequency responses were close, however our asymmetric transfer function at the first section of the preamp described in section 3.1, introduced more even harmonics than the real amp and guitar rig on crunch settings. Figure 9 shows the frequency response of an input signal and of the signal after the preamp stage (with the two wave shapers set with the transfer functions from Figure 5). In white we see the even harmonics and the peaks between them are the odd harmonics. This corresponds to a warm sound, rich in harmonics.

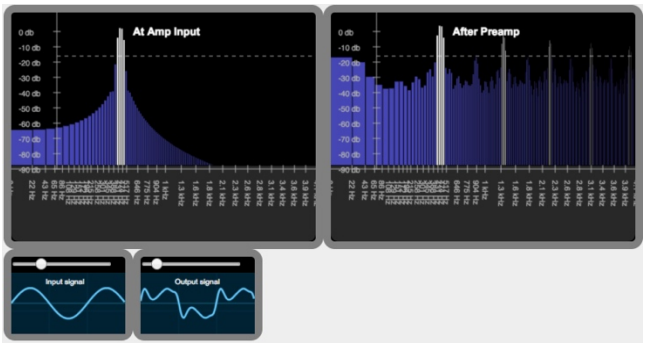

Figure 9: Frequency responses and a Sin wave at 440Hz. At input and after the preamp stage.

5. A few words about the GUI and MIDI support

The current version of the amp GUI relies on a set of Polymer web components called "webaudiocontrols[19]. They propose photo-realistic knobs, sliders, leds, switches and look like popular widgets from the native audio plugin world. The sprite sheet images they rely on have been created using an online generator (WebKnobMan[20]), a web based version of the popular KnobMan tool from the VST plugin scene. We contributed to this set of widgets by adding a midi learn context menu to all of them, enabling control from external MIDI devices. We used the WebAudio amp with both a Behringer FCB1010 MIDI pedalboard (for changing presets) and with a Novation Launch Control pad (that comes with a set of rotating knobs and buttons we mapped to the rotating knobs and switches of our GUI) [21], but it can be controlled using any MIDI device.

6. Blind tests and interviews

We asked four guitar players with different background to play different guitar amp simulations (Guitar Rig by Native Instruments -a reference used by many recording studios and musicians, the GarageBand JCM800 plugin, Guitarix JCM800 –an open source amp sim- tried on a dedicated Linux device: the Mod Duo pedal, and our WebAudio amp), as well as the Yamaha THR10 amplifier, that uses digital modeling too. We paired the amp sims with a speaker simulation and matched cabinet, and with a reverb when necessary, to have similar configurations (amp, reverb, cabinet simulation) that could be compared.

We used a Blade Austin guitar (two single and a double coil pickup + active circuitry for boosting pickups with a push pull knob), a Mac Book pro, the Presonus 44VSL sound card and a pair of Tapco S5 studio monitor speakers (see Figure 10). We used both Google Chrome and Firefox nightly. This is our lowest latency hardware configuration. We let our testers change pickups settings on the guitar, and we tweaked the amp simulations on demand ("please add some treble, add some reverb, give me more crunch…"). Our testers' experience is > 10 years of guitar playing, 3 of them played in bands for years, gave gigs, are used to guitar amps, 1 of them plays a lot with amp simulators at home.

Figure 10: Typical setup during qualitative evaluation

[18] http://wiki.moddevices.com/wiki/MOD_Duo

[11] https://www.presonus.com/community/Learn/The-Truth-About-Digital-Audio-Latency

[19] https://github.com/g200kg/webaudio-controls

[20] http://www.g200kg.com/en/webknobman/

[21] We also re-created a web version of these two MIDI devices, using custom made webaudio controls, for making the MIDI mapping process more ergonomic. See: https://wasabi.i3s.unice.fr/AmpSim3/midi.html

We asked them to rank the different amp sims using this scale: E (bad), D (average), C (good), B (very good) and A (Excellent). Tables 2-6 show the results. We also asked them to describe with a few words the sounds and the feeling of play (Table 7).

You can find videos and a complete report of these sessions online[22].

Table 2. "Can you feel the latency and is it annoying for playing comfortably?"

	Guitar Rig JCM800	Garage Band JCM 800	Guitarix JCM800	Web Audio Amp (Chrome/FF)
User 1	No	**Yes**	No	No/**Yes** in high notes but ok for playing
User 2	No	**Yes**	No	No/**Yes** in high notes but ok for playing
User 3	No	**Yes**	No	No
User 4	No	No	No	No

Table 3. "How can you qualify the sound (clean sound)?"

	Guitar Rig JCM800	Garage Band JCM 800	Guitarix JCM800	Web Audio Amp
User 1	A	D	E	A
User 2	B	C	D	A
User 3	A	D	D	C
User 4	D	D	D	C

Table 4. "How can you qualify the sound (crunch sound)?"

	Guitar Rig JCM800	Garage Band JCM 800	Guitarix JCM800	Web Audio Amp
User 1	B	E	E	B
User 2	B	C	D	C
User 3	B	D	E	B
User 4	D	D	D	D

Table 5. "How can you qualify the sound (distortion sound)?"

	Guitar Rig JCM800	Garage Band JCM 800	Guitarix JCM800	Web Audio Amp
User 1	A	E	E	B
User 2	B	C	D	C
User 3	B	D	E	B
User 4	D	E	E	D

Table 6. "How can you qualify the dynamic response?"

	Guitar Rig JCM800	Garage Band JCM 800	Guitarix JCM800	Web Audio Amp
User 1	A	D	E	B
User 2	B	C	D	C
User 3	C	B	D	B
User 4	D	D	D	C

Table 7. "Describe in a few words your feelings

	Guitar Rig JCM800	Garage Band JCM 800	Guitarix JCM800	Web Audio Amp
User 1	Very nice twang sound, we feel the tubes, excellent dynamics. Accurate distortion. Very comfortable.	Not bad, "round" sound, too greasy, something is missing.	Bad. This one sounds very digital, like in a closed box.	Nice, warm, clean sound, we feel the tubes. More structured distorted sound,
User 2	Tube feeling, good sound in each setting. Warm and responsive. Fells like a real amp.	Nice clean sound, tubes, distortion is not accurate enough. Latency is a problem	Strange feeling, very digital. Cold sounds and bad feeling.	Nice clean sound, tubes here. "humid" for rhythmic with distortion very good for soloing
User 3	Good feeling, good dynamics, good sound	Toy amp, sound not accurate, clean sound ok.	Sounds weird, digital feeling.	Warm clean/crunch sound, accurate distortion, nice. Good feeling
User 4	Sounds digital. Too much "round" sound.	Not enough nuances, not warm and versatile enough.	Sounds digital. Cold.	Not warm enough, dynamics ok, Good distortion.

Table 8. (switching to oversampling 2x and 4x) "Can you hear a difference in the sound?"

User 1	User 2	User 3	User 4
No	No	No	No

During the tests, we turned the oversampling mode of wave shaper nodes on and off with values equal to 2x and 4x, and asked our tester if they can hear or feel a difference. None of them noticed any change (Table 8). Also, Google Chrome and Firefox proposed a similar experience: two testers could not distinguish them with the WebAudio amp sim, and two noticed the latency on high notes (more with Firefox that has a higher latency than Google Chrome), but they said they could handle it and that it was not inconvenient.

User 4 did not like any of the amp sim he tried, he's used to play in the style of Jimi Hendrix on a real tube amp at home and felt that all simulations sounded "digital", but the Yamaha amp didn't. The WebAudio simulation is a close second to Guitar Rig, a reference in the audio plugin world, and this is very encouraging. The LV2 Guitarix JCM800 plugin running on the Mod Duo pedal did not sound right to any of our testers. Other plugins on this pedal sounds much better, so we wondered if the port of this software on the Mod Duo has been made correctly.

Our reference hardware amp, the Yamaha THR10, got an A in each category, with all our testers. Its speakers are not "neutral" like the studio monitor speakers we used, and they certainly add a lot to the sound color. However, the simulations we used during the evaluations are not are not yet comparable to dedicated DSP hardware..

7. Perspectives

This amp simulation is part of the ANR WASABI project (42 months, started in January 2017), that consists in building a 2 million song database with different client applications for music schools, sound engineering schools, musicians / composers, etc.

[22] https://wasabi.i3s.unice.fr/AmpSim3/userEvaluation.html

The amp is part of a future set of WebAudio applications that will include a guitar pedalboard with chainable effects (a prototype demo shown at the WebAudio conference 2016 [19] is already available, and includes a previous version of the amp presented in this paper[23]), a mixing table, sound analysis tools, and so on, linked and pre-tuned for classic songs from the database.

The amp model can be enhanced with a more faithful tone stack model, a better power amp model (we did not model yet the effect of speakers' impedance), tube feedback effect simulation could be improved by adding more dynamics (with an envelope follower at the input of the amp, that could drive the "strength" of the transfer functions), and when the AudioWorklet node will be available, more accurate simulation of triodes will hopefully be possible without adding too much latency. Other amp models are planned and will be developed during the project. The GUI needs also to be polished (there are too many "advanced settings" widgets and measurement tools for a normal user), and ergonomic tests should be conducted.

8. Conclusion

We could not find any previous work that reproduced each stage of a specific guitar tube amplifier in WebAudio, and we showed that it is possible to make a WebAudio based guitar amp simulation that is both playable "guitar in hands", and competitive in terms of sound and comfort of play with some reference native plugins. The use of simplified tube models based on wave shapers, while not as accurate as more advanced models based on differential equations, is a valuable solution for making a guitar tube amp simulation, when accompanied by filters and fine-tuned gain settings. Hopefully, the upcoming AudioWorklet node will make better modeling possible for both preamp / amp tube stages and for the tone stack.

However, the main problem that remains with WebAudio today, and illustrated by our measures and experiments, is still the lack of low latency audio driver support by standard browsers on Windows, the lack of a mean to set the audio buffer size or to set the sample rate to adjust the latency, and finally, the lack of support for multiple audio inputs/outputs. Jack Audio is certainly the way to go as it may act as a "proxy/barrier" between browsers vendors and licensing policies from audio driver publishers (ASIO for ex.).

9. ACKNOWLEDGMENTS

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

Acknowledgements also go to Ivan Cohen and Alain Poulin (aka LePou) for their encouragements and precious help in understanding how guitar amp internals work, providing literature, and giving their precious time for testing and sending feedback.

10. 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. 18th Int. Conference on Digital Audio Effects (DAFx-15), Trondheim, Norway, 2015.

[3] Cheng-Hao C. A Guitar Overdrive/Distortion Effect of Digital Signal Processing. https://tinyurl.com/kc6s4er

[4] Macak, J., & Schimmel, J. 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). 2010.

[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] Yeh, D. T., & Smith, J. O. (2006, September). Discretization of the'59 Fender Bassman tone stack. In *Int. Conf. on Digital Audio Effects (DAFx-06)* (pp. 18-20).

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

[8] Hotz, M. A Study of Tube Amplifier Modeling Using Nonlinear Wave Digital Filters. Master student thesis. https://tinyurl.com/kzhj32x

[9] Clark, J. J. (2003). Advanced programming techniques for modular synthesizers. https://tinyurl.com/3xskmv

[10] Millet P. "The sound of distortion", online tutorial. http://www.pmillett.com/file_downloads/ThesoundofDistortion.pdf

[11] Denton D., "Electronics for Guitarists", Springer Science & Business Media. 2013. (typ. Page 247)

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

[13] Donaldson, N.P, "An Electric Guitar Plucked String Model for Realtime Control with Distortion and Feedback"McGill University, MUMT 618 Final Project, Fall 2009.

[14] Zakai, A. 2011. Emscripten: an LLVM-to-JavaScript compiler. *ACM Int. Conf. companion on Object oriented programming systems languages and applications companion* (OOPSLA '11). ACM, New York, NY, USA, pp. 301-312. Website at http://kripken.github.io/emscripten-site/

[15] Adenot P. Web Audio API performance and debugging notes, 2nd WebAudio Conference, Atlanta, 2016. http://padenot.github.io/web-audio-perf/#latency

[16] Randall, A., Proper Dummy Load on Output of Tube Amp, 1998. https://tinyurl.com/myxo9k3

[17] Cohen, I., & Helie, T. (2009, October). Simulation of a guitar amplifier stage for several triode models: examination of some relevant phenomena and choice of adapted numerical schemes. In *Audio Engineering Society Convention 127*. Audio Engineering Society.

[18] G2 Workshops and tutorials : Waveshaping and Distortion. https://rhordijk.home.xs4all.nl/G2Pages/Distortion.htm

[19] Buffa M., Demetrio M., Azria N. Guitar pedal board using WebAudio. Web Audio Conference, Apr 2016, Atlanta, United States. 2016.

[20] D. Difilippo and K. Greenebaum. Audio Anecdotes: Tools, Tips, and Techniques for Digital Audio: Tools, Tips, and Tricks for Digital Audio: v. 1. Book. 2004.

[23] https://mainline.i3s.unice.fr/GuitarProcessor/