Can you DAW it Online?

The Challenges of a Web-Based Open Source Workstation

Michel Buffa
Université Côte d'Azur,
I3S, CNRS, INRIA
Sophia-Antipolis, France
michel.buffa@univ-cotedazur.fr

Samuel Demont
Université Côte d'Azur,
I3S, CNRS, INRIA
Sophia-Antipolis, France
samuel.demont@etu.unice.fr

***Abstract*— This article presents WAM-Studio, an open-source digital audio workstation (DAW) that leverages W3C standards such as Web Audio, Web MIDI, Web Assembly, and Web Components, as well as the Web Audio Modules (WAM) plugin standard (aka "VSTs for the Web"). Developed since 2022, WAM-Studio and its fork Attune have already been the subject of several publications. This paper focuses on performance (including latency management) and details the addition of new features, such as MIDI tracks that were not supported previously. Today, most online DAWs are closed-source, and the rare open-source initiatives are either unmaintained or less complete than the work presented in this article. WAM-Studio has been developed as an open-source demonstrator of Web standards and contains examples of design and implementation of functionalities that are complex to achieve correctly (audio and MIDI recording in synchronization with other tracks, offline rendering of a project, etc.) An online demo and a GitHub repository for the source code are available.**

Keywords—Online Digital Audio Workstation, Web Audio, Web MIDI, WebAssembly, Web Audio Plugins, Web Audio Modules

I. Introduction

Computer-Assisted Music (CAM) is a constantly evolving field that uses computers to record, edit and produce music. Digital Audio Workstations (or "DAWs") are software programs specially designed for CAM, enabling users to create and manipulate digital audio and MIDI content [17].

Audio plug-ins are software modules that add extra functionality to DAWs, giving users greater flexibility and control over their music production [16]. The DAW market began with the Atari ST (1985) - the first computer to support the MIDI standard - and Steinberg's Cubase DAW (1989). Shortly afterwards, the VST standard for audio plugins was proposed (1997), and since then thousands of plugins have been developed for use in the main native DAWs available on the market[1].

A DAW is a feature-rich piece of software, and therefore particularly complex to develop in terms of design, implementation and ergonomics. It enables the creation of multitrack pieces by directly using audio samples (e.g. by incorporating an audio file into a track or recording from a microphone or sound card input), mixing them, applying sound effects to each track (e.g. reverb, frequency equalization or auto-tune on vocals), but also by using tracks with virtual instruments (software reproduction of a piano, violin, synthesizer, drums, etc.). The track is then recorded in MIDI format in the form of events corresponding to the notes and additional parameters (such as the velocity with which the keys on a piano keyboard were pressed, for example).

These events enable the track to be played back by asking a virtual instrument to synthesize the signal. A DAW can therefore play back, record and mix both audio and MIDI tracks, edit these tracks (copy/cut/paste within a track or between tracks), manage real-time audio effects and virtual instruments, mix and export the final project in a simple format (e.g., a WAVE or MP3 file). In the native world, these effects and instruments are the "audio plugins" that extend the capabilities of standard DAWs. Since 1997, a significant market for third-party plug-in developers has developed. Four DAWs dominate the market (Logic Audio, Ableton, Pro Tools, Cubase[2]), and the existence of several proprietary plug-in formats complicates the task for developers.

Many of these modern Digital Audio Workstations (DAWs) offer robust collaborative features [3], allowing musicians and producers to work together seamlessly on projects, regardless of their physical location, but the candidates of choice are the online DAWs, that can run in web browsers and do not require any installation. Contrarily to native DAWs, the web-based DAW market is still emerging: the first online DAWs appeared in 2008, using Flash technology [15]. Those using HTML5 and the Web Audio API only appeared in the period between 2015 and 2016, as the technologies are much more recent (the first implementations of the Web Audio API in browsers date back to 2012). By 2023, several web-based DAWs are available, mainly commercial. An interoperable plugin format called Web Audio Modules (WAM) exists and is supported by at least one online DAW [9].

The previously mentioned features of online DAWs also make them a good fit for the Internet of Sounds [20].

In this paper we address research questions such as: "What are the main challenges in developing an online DAW?",

[1] https://www.musicradar.com/news/tech/a-brief-history-of-computer-music-177299
[2] https://www.mordorintelligence.com/industry-reports/digital-audio-workstation-market

[3] Ableton Live offers collaboration features like Link, which allows users to sync multiple instances of Live over a network (https://www.ableton.com/en/link/)

"What technical constraints do we have to deal with?", "How to deal with latency?", "Are current web standards adapted to developing a high-performance DAW?"

Section II details the audio technologies used in WAM-Studio, section III presents the related works, section IV focuses on the design and implementation of WAM-Studio, section V is about latency issues related to audio / MIDI playback and recording, section VI present other additional features that have been added to WAM-Studio and finally section VII concludes and presents future works.

II. Web Audio and Web Audio Modules

in 2021 the W3C Web Audio API became a "recommendation" (a frozen standard). It offers a set of "audio nodes" that process or produce sound, and these nodes are used from JavaScript code by instantiating classes provided by the API. These nodes can be connected to form an "audio graph". The sound moves through this graph at the sampling rate (default value 44100Hz, this value can be modified) and undergoes transformations [8]. Some nodes are wave generators or sound sources corresponding to a microphone input or a sound file loaded into memory, others transform the sound. These nodes are connected to the browser via JavaScript, enabling a wide range of different applications involving real-time audio processing to be designed. It is not just music applications that require complex audio processing; the API is also designed to meet other needs, such as video game development, multimedia, video conferencing and so on. The API comes with a limited set of "standard" nodes for common operations such as gain control, frequency filtering, delay, reverb, dynamics processing, 2D and 3D sound spatialization, etc.

In general, audio libraries/APIs utilize a control thread and a rendering thread for their operations ([2], [3], [4]). The Web Audio API follows this model with a rendering thread, known as the "audio thread," dedicated to processing sound through the audio graph and delivering samples to the operating system for playback. This thread operates under strict real-time constraints and high priority to avoid audible glitches if it fails to deliver audio samples on time (128 samples by default). The control thread handles the user interface's JavaScript code, calls to the Web Audio API, and modifications to the audio graph, such as connecting/disconnecting nodes and adjusting parameters. Standard nodes in the Web Audio API are instances of subclasses of the AudioNode class, providing predefined processing coded in C++ or Rust (for Firefox) that runs in the audio thread. These nodes' algorithms cannot be modified, only their parameters can be adjusted via JavaScript. The exception is the AudioWorklet node, added in 2018 [5], which allows for custom low-level audio processing in the audio thread, albeit with constraints such as no asynchronous operations, file imports, or access to the HTML page DOM.

Developers can assemble Web Audio API nodes into a graph to create complex audio effects or instruments, such as delay effects, auto-wah, chorus, distortion, synthesizers, and samplers. DSP code from languages like C/C++ or domain-specific languages like FAUST can be compiled into WebAssembly and executed in a single AudioWorklet node.

This approach has led to the development of many high-level audio effects and instruments. However, chaining these effects and instruments, as is common in guitarist pedalboards or DAWs, can be cumbersome with the low-level Web Audio API nodes. This situation highlighted the need for a higher-level unit to represent native audio plugins on the web. Until 2015, there was no high-level standard for "audio plugins" and "host" applications on the Web platform. Since then, several initiatives have emerged, with one becoming a "community standard," which is detailed in the next section.

Web Audio Modules

In 2015, Jari Kleimola and Olivier Larkin proposed "Web Audio Modules" (WAM) as a standard for Web Audio plugins [11]. Kleimola then helped create one of the first online DAWs, ampedstudio.com, using WAMs. In 2018, researchers and DAW industry developers extended the original WAM project, resulting in a more versatile standard [8]. Web Audio Modules version 2.0, released in 2021, established a community standard (API, SDK) that supports various build tools, TypeScript, and frameworks like React, improving performance and simplifying plugin parameter access and DAW integration [13]. This version features an innovative design for managing communication between plugins and host applications without relying on the Web Audio API's parameter management.

This design ensures high performance when both a DAW and plugins use AudioWorklets. This use case was not considered when the Web Audio API was designed, necessitating this new approach. The WAM standard allows indeed communication to remain within the high-priority audio thread[4], essential for efficient audio processing in DAWs and plugins built with AudioWorklet nodes. Example: during playback, a MIDI track sends notes to a virtual instrument plugin and modifies its parameters at the sample rate (a-rate). A DAW can have multiple tracks, each with numerous plugins and parameters. The WAM standard detects when both DAW and plugins are AudioWorklets and optimizes communication using shared memory and a ring buffer, avoiding the need to cross the audio thread barrier. This eliminates the need for sending events from the control/GUI thread, which would be necessary with the Web Audio API's parameter management. In summary, the WAM standard simplifies plugin and host application creation and ensures optimal communication between hosts and plugins.

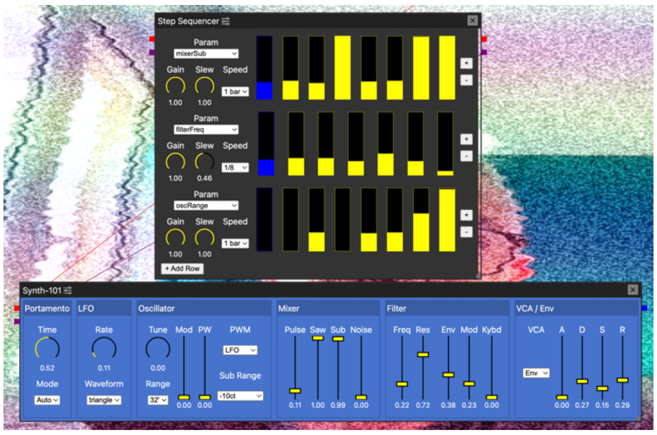


Figure 1: Same component (programmed differently) from previous picture, can modulate parameters of a WAM synthesizer.

Dozens of WAM plugins are available through the WAM Community Endpoint and can be easily integrated into host

[4] When the target WAM plugin is itself AudioWorklet-based.

applications using their URI[5]. We are talking here about elements comparable in quality and complexity to those existing in the world of native applications (some instruments and effects are in fact Web Assembly ports of the source code of true native plugins written in C++). The WAM ecosystem is evolving rapidly, for example, as an extension system is now supported [10], allowing WAMs to manipulate video input and outputs, to do 3D rendering or WebGL shader animations, to support parameter modulation (WAMs that modulate other WAM parameters) or to load/save assets from the host storage (i.e. useful for samplers). Available WAMs [10] include *note generators* such as: a piano roll, a programmable step sequencer, random note generators, chord generators, *virtual instruments*: synthesizers (Figure 1), samplers (Figure 2), physical modeling of instruments (flute, clarinet, brass, djembe), *audio effects* (more than one hundred available, including all classics: flanger, chorus, reverb, distortion, fuzz, overdrive, envelope followers, etc.).

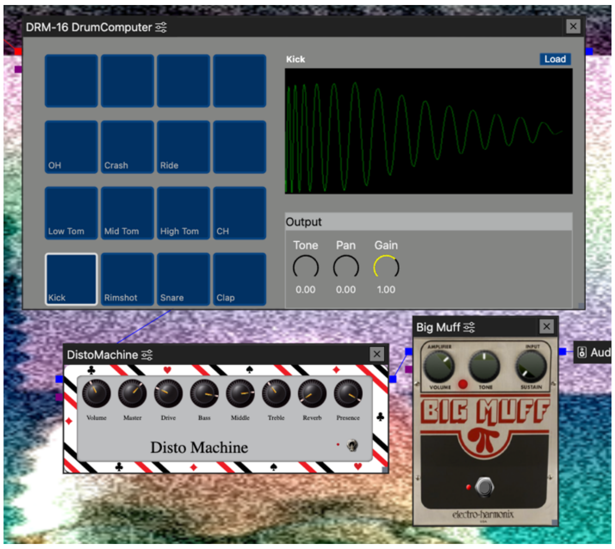


Figure 2: a drum sampler, and two distortion effect WAMs.

III. Related Works

The first online DAWs based on the Web Audio API appeared between 2015 and 2018, including Audiotools, BandLab, Amped-, SoundTrap [18], and Soundation. These DAWs share several characteristics: they are commercial, closed-source, and have limited academic coverage. They facilitate remote collaboration on musical projects, a feature that gained popularity during the COVID-19 pandemic (2021-2022). Collaboration modes vary from synchronous (like Google Docs) to asynchronous (through link sharing), and communication tools like chat and video conferencing are often included. All these DAWs offer typical functionalities such as audio and MIDI recording, track editing, mixing, and support for effects and virtual instruments. They differ mainly in ergonomics and their design focus, with some optimized for PCs and others for mobile devices. See [15] for a complete survey of online DAWs. To date, AmpedStudio is the only DAW to support an open plug-in standard: Web Audio Modules.

When it comes to open-source software, there are few options. GridSound[6], a DAW developed since 2015, supports audio and MIDI, but offers only basic effects and instruments and does not support plugins. It no longer appears to be actively maintained. Other popular open-source JavaScript libraries for multitrack players/recorders include wavesurfer.js, peaks.js, and waveformplaylist[7], as well as Audacity-style audio editors such as AudioMass[8]. These projects do not use low-level custom DSP processing (no AudioWorklet), do not optimize DAW/plugin communication and do not support external plugins. To overcome these limitations, WAM-Studio has been developed to showcase these advanced techniques.

IV. WAM-Studio: Design and Implementation

WAM-Studio is an online tool for creating audio projects, designed as multi-track musical compositions. Each track represents a different layer of content that can be recorded, edited, played back or integrated with audio files. Some tracks can control virtual instruments, containing only the notes to be played and metadata. Users can add or delete tracks, play them individually or together, and arm them for recording. During recording, all other tracks play simultaneously, while armed tracks record new content.

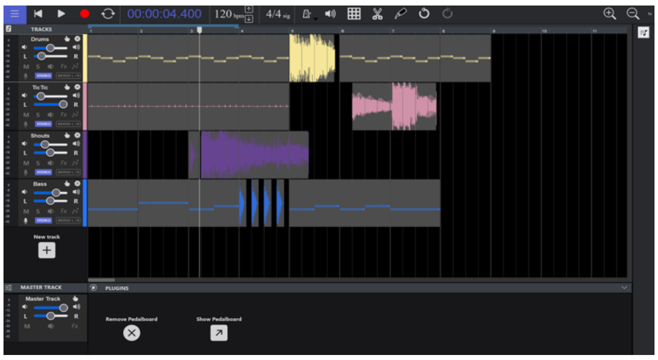


Figure 3: Tracks in WAM-Studio. Regions can be audio or MIDI and can overlap, be moved, deleted, split, merged, etc.

Tracks

In WAM-Studio, each track is a container for audio or MIDI related data, accompanied by an interactive representation of this data, editing and processing functions, and a few default parameters such as the track's volume and left/right panning. Figure 3 shows some audio tracks in WAM-Studio with the associated audio buffer region (waveforms) and MIDI regions (squares) displayed and the default track controls/parameters on the left (mute/solo, record arming, volume, stereo panning, automation curve display, effects plugins). As many tracks can be displayed, scrolled during playback, zoomed and edited, we used the pixi.js library to efficiently manage drawing and interaction within an HTML canvas. This library uses GPU-accelerated WebGL rendering and offers many features for managing multiple layers on a single HTML5 canvas (Figure 3).

[5] The WAM Community plugins and API is available from the https://www.webaudiomodules.com/ web site.
[6] https://gridsound.com
[7] https ://github.com/naomiaro/waveform-playlist
[8] https://audiomass.co

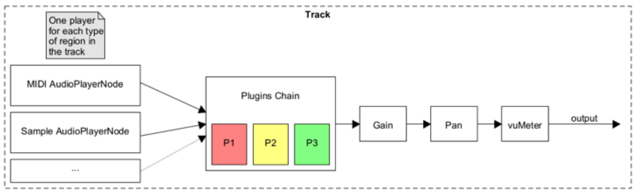


Figure 4: Audio graph of a track implementation.

Figure 4 shows the audio graph corresponding to the processing chain of an audio track. The sound goes from left to right: first the "track player/recorder/editor" is implemented as an AudioWorklet node, using custom code to render an audio buffer or a MIDI region, then the sound goes through a chain of WAM plugins for adding audio effects then the output signal has its gain and stereo pan adjusted, then we have another AudioWorklet node for rendering volume in a canvas (VU-meter).

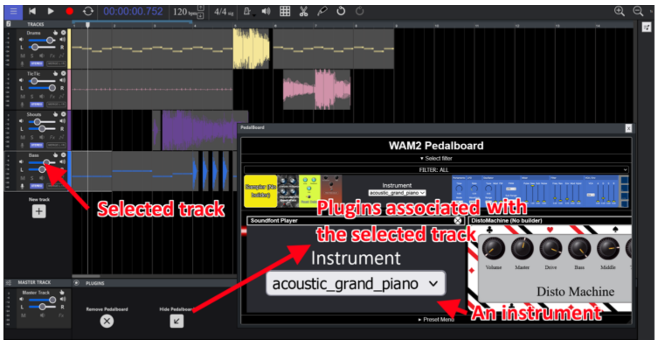


Figure 5: WAM plugins associated with a track.

Figure 5 shows that each track can also be associated with a set of plugins to add audio effects or generate music (in the case of instrumental plugins). In most DAWs, tracks are typically categorized into "audio" tracks, which contain recorded audio like vocals or instruments represented by waveforms, and "MIDI" tracks, which hold MIDI data controlling virtual instruments without audio information. MIDI tracks can be edited in a matrix display to modify MIDI events and are associated with virtual instrument plugins. However, some DAWs, like AmpedStudio prefer "hybrid" tracks that can contain both audio and MIDI regions. WAM-Studio *supports the hybrid track approach* which offers greater flexibility in composition, simplified implementation and better performances (the AudioWorklet associated with a track will take care of all host-plugin communications without leaving the audio thread).

In recording software like WAM-Studio, each track feeds into a "master track", where users can adjust overall volume and stereo panning for the final mix. The master track can also host a chain of WAM plug-ins to adjust dynamics and final frequencies, offering precise control for final mastering steps. All effects and virtual instruments are WAM plug-ins, allowing their parameters to be automated. This configuration makes it possible to produce complex audio productions entirely in a web browser, taking advantage of extensive control and adaptability. During playback, DAW tracks are transformed into audible audio signals, which are combined with the "mix" when the play button is pressed. Offline rendering options allow the final mix or individual tracks to be exported to the local filesystem (as.wav of .mp3 files), using the Web Audio API's OfflineAudioContext. WAM-Studio is the only open-source software to integrate these advanced features, so its source code can be very useful for web audio developers to study.

Tracks are WAM Processors (player / MIDI renderer in the audio thread)

The Web Audio API provides the `AudioBufferSourceNode` which facilitates basic playback of stored audio buffers and could have been used for the implementation of Audio tracks. However, for DAWs requiring precise automation of parameters during multitrack playback and recording, a more sophisticated approach is required. This involves interpolating plug-in and track parameters at high frequencies (a-rate or k-rate[9]), potentially managing many parameters across different plug-ins and tracks. To achieve this level of control, WAM-Studio opts for a custom-designed low-level audio player instead of the standard AudioBufferSourceNode. Each audio track is structured as an "AudioWorklet processor", taking advantage of the Web Audio Modules SDK to execute custom DSP code in the audio thread with high priority. This processor, which inherits from the WamProcessor class (from the WAM SDK), extends the AudioWorkletProcessor class and implements a `process(input, output)` method called at sample rate. Here, sound samples are processed, audio buffers are rendered, and data is exchanged with WAM plugins - including MIDI events for virtual instruments - thanks to efficient communication methods provided by the inherited WamProcessor class. This approach gives WAM-Studio extensive control over playback and recording behavior, which is essential for managing the automation of complex parameters seamlessly in the DAW environment.

The Web Audio API employs AudioWorklets, consisting of two main components: AudioWorkletNode and AudioWorkletProcessor, which operate in separate threads—GUI thread and audio thread respectively. WAM-Studio, as a WAM host application, implements tracks as subclasses of `WamProcessor`, facilitating precise control over audio processing. WAM plugins associated with these tracks can utilize WamNode and WamProcessor classes for their DSP core too, though it is not mandatory: they can only be built of high-level audio nodes from the Web Audio API. Regardless of implementation, the WAM design ensures seamless communication between hosts and plugins, optimizing performance with shared memory techniques like Shared Array Buffers for efficient data exchange, when possible (WAM plugins are AudioWorklet/WamProcessor based), or be less optimal and involve crossing the thread barrier. For managing plugin chains, WAM employs a singleton object called WamEnv, facilitating interactions across threads and encapsulating SDK versions to ensure compatibility and security.

Each plug-in group (WamGroup) comprises the plug-ins created by a specific host, and it is possible to send events downstream (MIDI, for example) to all plug-ins in a chain. A considerable effort has been made to reduce the number of assumptions made by the WAM API regarding host

[9] *A-rate* means "audio-rate" and indicates that the processing is done at the sample rate of the project, typically 44100 times/second. *K-rate* means "kilo-rate" and indicates that the processing is done at a relatively low frequency, typically once per audio block (at 44.1Khz, with 128 audio sample blocks, this means every 3ms).

implementation, with WamEnv and WamGroup instances being the only objects in the WAM system that are provided by the host. WamEnv is allocated at DAW creation and WamGroup instances are associated with each track.

WAM-bank: a plugin manager for each track

Plugin chains are managed using a special WAM plugin that also acts as a "mini host" (Figure 6). We call it the WAM-bank (or the "WAM pedalboard") [6]. It connects to one or more plugin servers, which return(s) the list of available plugins as a JSON array of URIs (a WAM plugin can be loaded simply using a dynamic import and its URI, see [9]). From this list of URIs, WAM plugin descriptors are retrieved, which contain metadata about the plugins: name, version, provider, thumbnail URI, type (effect or instrument), available inputs and outputs etc. When the pedalboard plugin is displayed in the DAW, the chain of active plugins is empty, and plugins can be added to the processing chain, deleted, reordered and their parameters set.

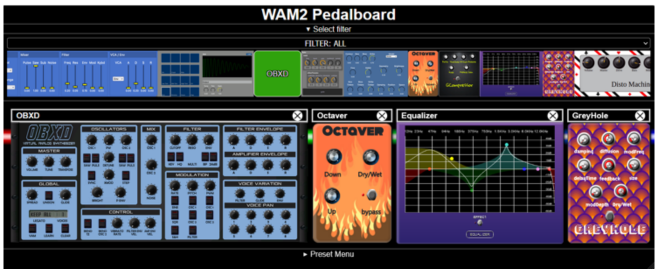


Figure 6: The WAM-bank plugin that manages chains of plugins (virtual instruments and audio effects) associated with each track.

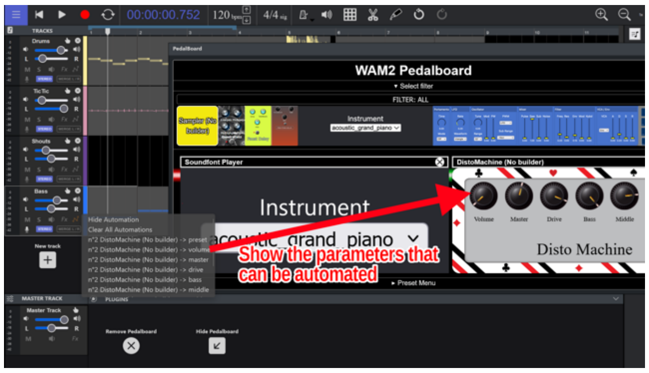


Figure 7: Parameter Automation menu updates when the chain of plugins is modified.

Any configuration can be saved as a named preset (e.g. "crunch guitar sound 1", "Synthesizer with ambient audio effects"). Presets can be organized into sound banks ("rock", "funk", "blues"). Managing the organization and naming of banks and presets is the responsibility of the WAM-bank plugin. The parameters exposed by this plugin correspond to all the parameters of the active preset (i.e. the sum of the parameters of the preset's plugins in the chain) and can be automated by the DAW. The WAM API allows events to be sent from plugins to the host when changes occur in the plugin configuration (in our case, when plugins are added or deleted in the current chain), and a parameter automation menu in each track GUI is automatically updated (Figure 7).

V. Dealing with Latency

1 - Audio track recording

Figure 8 shows the workflow / audio graph corresponding to the track recording process: first we need to obtain a multimedia stream from the user's microphone or sound card input using the MediaDevices API. Then we can use a MediaStreamSourceNode from the Web Audio API passing the multimedia stream to its constructor. This is how we expose a live audio stream to the Web Audio API. Secondly, we need a way to store this stream in a buffer. This can be done using the W3C MediaRecorder API or using a low-level AudioWorklet-based solution. The advantage of the MediaRecorder solution is its simplicity, but all tests carried out have shown that it is an unreliable solution, as the time required to allocate audio buffers and to start recording is unpredictable.

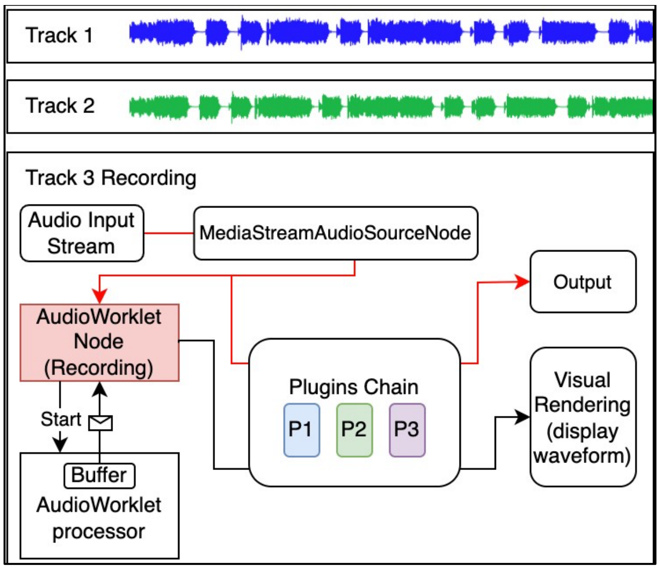


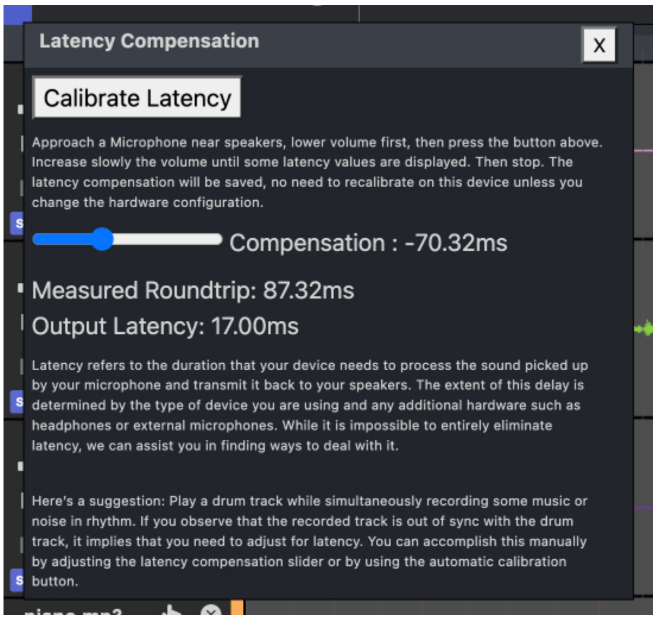
Figure 8: Workflow of the audio recording process.

We finally opted for a more accurate low-level solution, using a WamProcessor running in the audio thread whose role is to continuously record sound samples into a rotating buffer containing roughly one second of digitized sound. This buffer is in a memory shared with a JavaScript Worker running in a lower-priority thread, whose role is to consume the samples sent by the WamProcessor and to append to a buffer that grows as it goes along. We have here a classic producer/consumer design pattern, except that the producer is executed in the audio thread (and therefore non-interruptible, with high priority) and the consumer is in the graphic interface thread, with lower priority. This approach avoids memory allocations in the audio thread and is also more robust against possible CPU stress. The Worker also sends the current audio buffer as a message to JavaScript code running in another GUI thread. The latter can then draw the waveform of the recorded audio signal, which grows visually as the recording progresses. When recording is stopped, the track is ready to be played, since the audio buffer has been updated as it goes along.

Latency compensation: the recording process with a DAW is more constrained and structured than with a simple audio memo recorder, because we must take different types of latency into account:

- *The Input latency* is the time between the capture of an audio signal by the input device (sound card) and its processing by the audio context. It depends on the operating system, the configuration of the input device, the processing time of the audio system and the size of the buffer used by the audio context.
- *The Output latency* represents the time between the generation of a sound and its perception by the user.
- *The Round-Trip latency* is the sum of these two latencies and can be measured using an external recording device with two microphones: one on the physical sound source (for example, on the body of a guitar plugged into a sound card), and one in front of the speakers. Strike the guitar body, record the output and compare the input and output signals. We have carried out such measurements in the past [13]. It can also be estimated with a program emitting sounds (usually metronome-type sounds) and recording those same sounds by placing a microphone in front of the speakers.

Paul Adenot has proposed the implementation of a tool to measure this latency and the output latency[10]. The result is generally less accurate than when measuring with external tools. The input latency, crucial for real-time audio applications like recording guitar tracks alongside other instruments, can be derived by subtracting output latency from the measured round-trip latency. *Minimizing the round-trip latency is essential to ensure real-time synchronization and responsiveness for musicians during live performance and recording sessions*.


Latency Compensation

Calibrate Latency

Approach a Microphone near speakers, lower volume first, then press the button above. Increase slowly the volume until some latency values are displayed. Then stop. The latency compensation will be saved, no need to recalibrate on this device unless you change the hardware configuration.

Compensation : -70.32ms

Measured Roundtrip: 87.32ms

Output Latency: 17.00ms

Latency refers to the duration that your device needs to process the sound picked up by your microphone and transmit it back to your speakers. The extent of this delay is determined by the type of device you are using and any additional hardware such as headphones or external microphones. While it is impossible to entirely eliminate latency, we can assist you in finding ways to deal with it.

Here's a suggestion: Play a drum track while simultaneously recording some music or noise in rhythm. If you observe that the recorded track is out of sync with the drum track, it implies that you need to adjust for latency. You can accomplish this manually by adjusting the latency compensation slider or by using the automatic calibration button.

Figure 9: Automatic estimation of latency compensation.

Figure 8 illustrates the signal path for monitoring during recording, highlighted in red: part of the signal travels through the plugin chain and is audible (wet route), while the unprocessed signal is recorded (dry route, to the AudioWorklet). On macOS, using a buffer size of 128 samples (default in AudioContext), round-trip latency measures approximately 23ms in Chrome and 15ms in Firefox[11]. These latencies are comfortable and comparable to those achieved in similar scenarios with native applications like Logic Audio using guitar amplifier simulation plugins, even during fast guitar performances [13].

Nevertheless, when the newly recorded track is played back simultaneously, a delay of several milliseconds is visible and audible, the newly recorded track is "behind" the others. It is necessary to shift the new track backwards in time. This operation is called *latency compensation*, and the compensation value is equal to the input latency value. We have seen that this value can be calculated if the round-trip latency is known (and that the output latency is provided by AudioContext). In WAM-Studio, a configuration option automatically calculates the latency compensation by bringing a microphone close to the loudspeakers and initiating the calibration process[12] (the DAW emits sounds, records the result and compares emission and reception times). For a given hardware configuration, this operation only needs to be performed once (see Figure 9). Other classic optimizations have been implemented in WAM-Studio: as soon as the tracks are armed for recording, we allocate the ring buffer, start the recording Workers (JavaScript background tasks), pre-allocate the audio buffers and only start recording when the "record" then "play" buttons are pressed, at which point the actual recording begins. This avoids wasting time at start-up, with allocations and launching Workers.

2 - MIDI track playback and recording

For MIDI playback, as explained in section 0, each track communicates with plugins from the audio thread. If the plugin is implemented as an Audio Worklet Processor, then all communication occurs in this high priority thread and no memory allocation is required. If the plugin is not AudioWorklet based, then the WAM SDK will send a MIDI event that will be processed by the instrument plugin in the main thread, involving some latency. The performance evaluation has been conducted on a Mac Book Pro from 2024, using a MIDI keyboard and a digital recorder with two microphones bounded to the left and right channels. One microphone recorded the press on a key while the other recorded the resulting sound from the speakers. Measurements showed that even in this worst case, the latency is minimal and does not hurt playback (i.e. we used an audio track with accurate kick drum sounds at the beginning of each bar step, and played MIDI notes in another track, at the same position. No delay could be heard). The round-trip latency was measured down to 8ms on average (min 6ms, max 12ms), as we do not have a value for the output latency, we cannot estimate accurately the input latency. MIDI recording has been implemented in a similar way to audio recording: the host listens to MIDI events in the main thread[13], posts the MIDI data to a JavaScript worker which communicates with the Track AudioWorklet processor using a Shared Array Buffer and a ring buffer acting as a Single Producer / Single Consumer wait-free queue. The Track Processor sends the MIDI events to the virtual instrument(s) associated with the track, located in the WAM-bank plugin chain, for monitoring

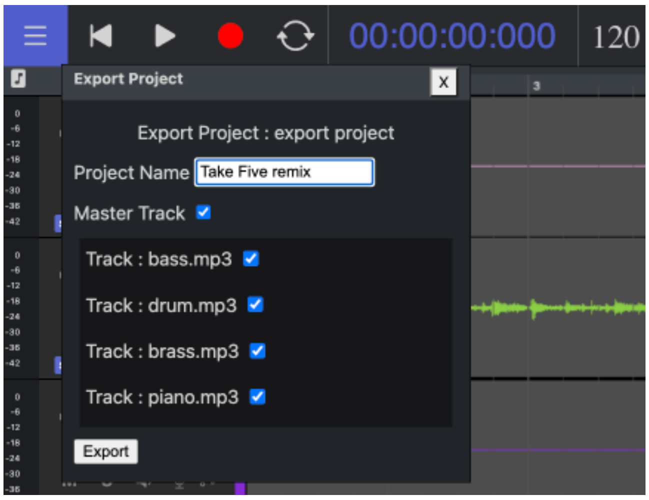
[10] https://blog.paul.cx/post/audio-video-synchronization-with-the-web-audio-api/
[11] Details about latency measurements from 2023 and 2018: https://mainline.i3s.unice.fr/jaes2023/latency.html
[12] Demonstration in this video: https://www.youtube.com/watch?v=CAAWoTjjlB0
[13] The Web MIDI API does not allow listening to events in an Audio Worklet, see discussion with Web MIDI implementers here: https://github.com/WebAudio/web-midi-api/issues/99

(a MIDI keyboard player must hear the “wet sound”). This communication stays in the audio thread if possible (see section 0). In our tests the “MIDI input latency” while not measurable, was neglectable, and so far, *MIDI latency compensation has not been necessary.*

VI. WAM-Studio other features

As shown in Figure 10, WAM-Studio allows you to export the final mix (master track) or each track individually (or both). The use of an `OfflineAudioContext` is used here to keep rendering time as short as possible. Effects chains and automation of their parameters are also considered. Possible formats are .wav or .mp3.

An audio loop browser has recently been integrated, using the WAM asset extension offered by Web Audio Modules, enabling the audio assets to be managed by the host and made available to plugins (Figure 11).


Export Project

Export Project : export project

Project Name: Take Five remix

Master Track ✓

Track : bass.mp3 ✓

Track : drum.mp3 ✓

Track : brass.mp3 ✓

Track : piano.mp3 ✓

Figure 10: Rendering project or individual tracks.

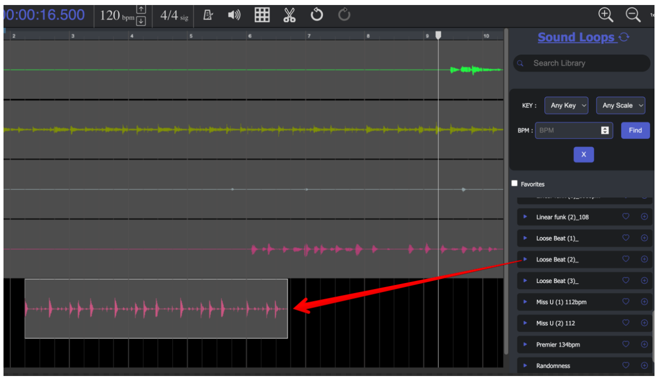


Figure 11: Audio loop browser.
Loops can be dragged and dropped into tracks.

The audio loop explorer offers a classic interface seen in native DAWs, with filters by instrument, tempo and key. Simply drag and drop loops onto a track to add an audio region to it. You can also drop a loop under the last track to create a new track. Note that drag and drop of external files is also supported, with the same behavior (region or new track created).

VII. Conclusion and future work

WAM-Studio is open-source[14], and this opened new possibilities! It has also given birth to Attune [15], a special version adapted for a research project in collaboration with the MERI team at Stanford's CCRMA, for the study and facilitation of multi-track music listening by hearing-impaired people fitted with cochlear implants. This version of WAM-Studio comes with a simplified GUI and a powerful macro system enabling the control of multiple plugins and plugin parameters with single widget controllers. Attune is online[15] and available in the different branch of the WAM-Studio GitHub repository[16].

WAM-Studio is a powerful online application that utilizes cutting-edge technology to enable users to playback, record audio and MIDI tracks, employ high-quality plugins (effects and virtual instruments), manage latency, and perform offline rendering. The source code is readily available (mono repository with front-end and back-end, including a simple Docker image for deployment). The project is developed in TypeScript, deliberately avoiding external frameworks, with the aim of making the code accessible to a wider audience and ensuring its long-term viability (minimal build tools). For those with a keen interest, WAM-Studio is like an "alarm clock to be disassembled" and will reveal the secrets of its design and implementation to the most curious, potentially uncovering insights into tasks that are not well-documented within the Web Audio and Web MIDI communities. Furthermore, the project serves as a compelling demonstration of the WAM plugin standard and its vast potential.

While WAM-Studio still lacks some classic features, such as dedicated editors for fine-tuning audio and MIDI regions, the most important goal remains the implementation of secure user account management and the collaborative aspects that we envision. Two other projects have shown that the WAM standard is particularly well-suited for the development of real-time collaborative host software (sequencer.party [10], another project within the Musical Metaverse [19], both based on state synchronization using CRDT-type algorithms).

In addition, as part of the French ANR research project DOTS, supported by IRCAM, the Web Audio API has been reimplemented to run outside the browser in devices such as Raspberry PIs [22], and we've run some WAM instruments and effects on these devices, making it possible to create smart WAM-based instruments [21].

We also plan to conduct a user-testing campaign to verify the validity of these development efforts for the Internet of Sounds community [20].

Acknowledgment

We would like to thank Antoine Vidal-Mazuy for his investment in WAM-Studio over the course of 2023 (he was its main coder and designer) and the team behind the Web Audio Modules, without which this DAW would never have existed.

This work has been supported by the French government, through the France 2030 investment plan managed by the *Agence Nationale de la Recherche*, as part of the "UCA DS4H" project, reference ANR-17-EURE-0004

[14] Online version : https://wam-studio.i3s.univ-cotedazur.fr/, source code : https://github.com/Brotherta/wam-studio

[15] https://attune.i3s.univ-cotedazur.fr/

[16] https://github.com/Brotherta/wam-studio

References

[1] M. Buffa, J. Lebrun, J. Kleimola, O. Larkin & S. Letz. Towards an open Web Audio plugin standard. In Companion Proceedings of the The Web Conference 2018 (pp. 759-766), 2018.

[2] F. Déchelle, R. Borghesi, M. De Cecco, E., B., and N. Schnell. jMax : an environment for real-time musical applications. In Computer Music Journal 23, 3, pp 50–58. 1999.

[3] J. McCartney. Rethinking the computer music language: Super collider. In Computer Music Journal 26, 4, pp 61–68. 2002.

[4] M. Puckette. FTS : A real-time monitor for multiprocessor music synthesis. In Computer music journal 15, 3, pp 58–67. 2002.

[5] H. Choi. Audioworklet : the Future of Web Audio. In Proceedings of the International Computer Music Conference. Daegu, South Korea, pp 110–116. 2018

[6] M. Buffa, P. Kouyoumdjian, Q. Beauchet, Y. Forner, and M.Marynowic. Making a guitar rack plugin -WebAudio Modules 2.0. In proceedings of the Web Audio Conference 2022., Cannes, France, 2022. https://doi.org/10.5281/zenodo.6769098

[7] J. Kleimola and O. Larkin. Web Audio Modules. In Proceedings of the Sound and Music Computing Conference 2015.

[8] M. Buffa, J. Lebrun, J. Kleimola, O. Larkin, and S. Letz. 2018. Towards an open Web Audio plugin standard. In WWW2018 – TheWebConf 2018 : The Web Conference, 27th International World Wide Web Conference. Lyon, France. 2018.

[9] M. Buffa, S. Ren, O. Campbell, J. Kleimola, O. Larkin, and T. Burns. 2022. Web Audio Modules 2.0 : An Open Web Audio Plugin Standard. In WWW '22 : The ACM Web Conference. 2022. ACM, France, pp 364–369. 2022 https ://doi.org/10.1145/3487553.3524225

[10] M. Buffa, S. Ren, T. Burns, A. Vidal-Mazuy & S. Letz. Evolution of the Web Audio Modules Ecosystem. In proceedings of the Web Audio Conference 2024. https://zenodo.org/doi/10.5281/zenodo.10825646

[11] B. Lazarevic and D. Scepanovic. *Unrevealed Potential in Delivering Distance Courses: the Instructional Value of Audio.* In eLearning and Software for Education. 2010..

[12] F. Lind and A. MacPherson. Soundtrap: *A collaborative music studio with Web Audio*. In Proceedings of the Web Audio Conference 2017. Queen Mary University of London, London, United Kingdom. 2017.

[13] M. Buffa and J. Lebrun. Real time tube guitar amplifier simulation using WebAudio. In Proceedings of the Web Audio Conference. Queen Mary University of London, London, United Kingdom. 2017.

[14] M. Buffa, A. Vidal-Mazuy, L. May, & M. Winckler. WAM-Studio: A Web-Based Digital Audio Workstation to Empower Cochlear Implant Users. In IFIP Conference on Human-Computer Interaction (pp. 101-110). Cham: Springer Nature Switzerland. 2023.

[15] M. Buffa, & A. Vidal-Mazuy. WAM-studio, a Digital Audio Workstation (DAW) for the Web. In Companion Proceedings of the ACM Web Conference 2023 (pp. 543-548). 2023.

[16] V. Goudard, and R. Muller. "Real-time audio plugin architectures." Comparative study. IRCAM-Centre Pompidou. France (2003).

[17] M. Marrington, 'Composing with the Digital Audio Workstation', in J.Williams and K.Williams (eds), The Singer Songwriter Handbook (New!York:Bloomsbury Academic, 2017), pp. 77-89.

[18] F. Lind and A. MacPherson. 2017. Soundtrap: A collaborative music studio with Web Audio. In Proceedings of the Web Audio Conference. Queen Mary University of London, London, United Kingdom.

[19] M.Buffa, D.Girard and A.Hofr. "Using Web Audio Modules for Immersive Audio Collaboration", IEEE Internet of Sounds Conference, Erlangen, Germany, 2024.

[20] L. Turchet, M. Lagrange, C. Rottondi, C. G. Fazekas, N. Peters, J. Østergaard, F. Font, T. Bäckström and Fischione, 2023. The internet of sounds: Convergent trends, insights, and future directions. IEEE Internet of Things Journal, 10(13), pp.11264-11292.

[21] L. Turchet,. Smart musical instruments: vision, design principles, and future directions, IEEE Access 7 (2019) 8944–8963.

[22] B. Matuszewski, and O. Rottier. The Web Audio API as a standardized interface beyond Web browsers. Journal of the Audio Engineering Society, 71(11), 2023, pp.790-801.

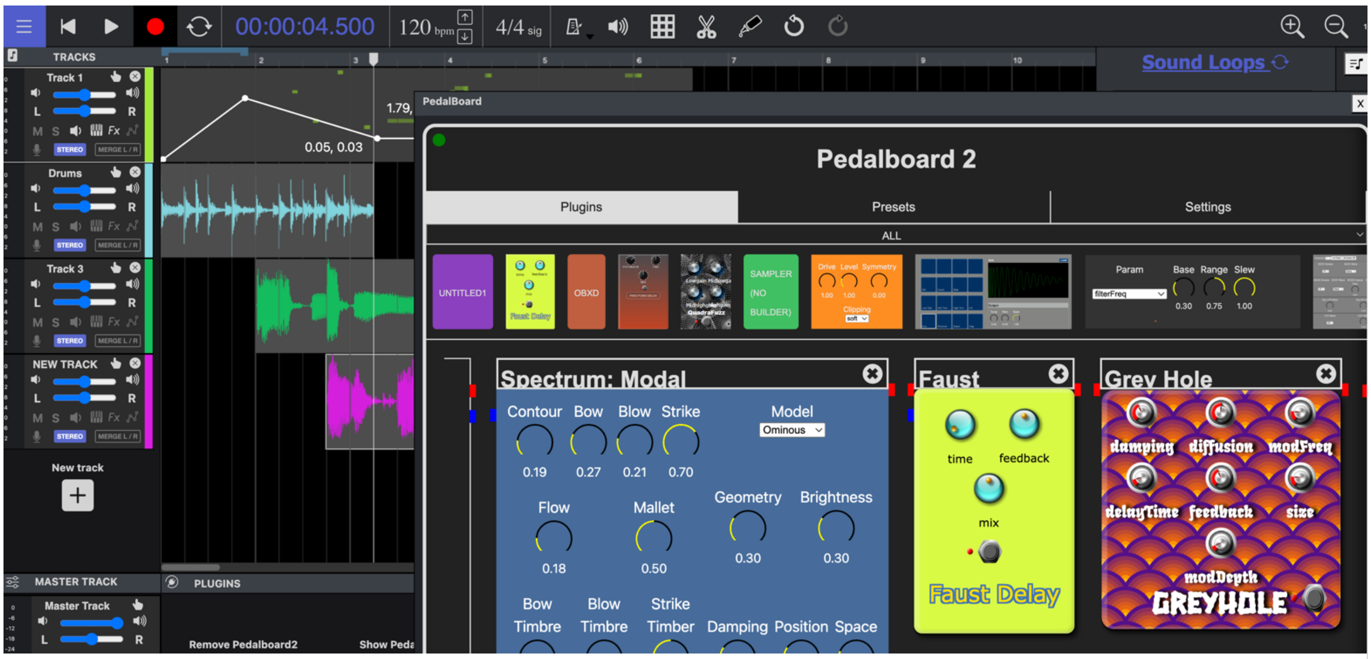


Figure 12: WAM-Studio typical project with MIDI and audio tracks, virtual instruments and effects associated to a track.