



HAL
open science

Une Station de Travail Audio-Numérique pour la Plate-Forme Web

Antoine Vidal-Mazuy, Michel Buffa

► **To cite this version:**

Antoine Vidal-Mazuy, Michel Buffa. Une Station de Travail Audio-Numérique pour la Plate-Forme Web. Journées d'Informatique Musicale 2023, May 2023, Paris, France. pp.8. hal-04334794

HAL Id: hal-04334794

<https://inria.hal.science/hal-04334794>

Submitted on 11 Dec 2023

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.



Distributed under a Creative Commons Attribution 4.0 International License

UNE STATION DE TRAVAIL AUDIO-NUMÉRIQUE POUR LA PLATE-FORME WEB

Antoine Vidal-Mazuy

University Côte d'Azur, CNRS, INRIA
antoine.vidal-mazuy@etu.univ-cotedazur.fr

Michel Buffa

University Côte d'Azur, CNRS, INRIA
buffa@univ-cotedazur.fr

RÉSUMÉ

Cet article présente WAM Studio (Figure 1), une station de travail audio numérique (DAW) en ligne open source qui tire parti de plusieurs APIs et technologies standards du W3C, telles que Web Audio, WebAssembly, Web Components, Web Midi, Media Devices, etc. WAM Studio s'appuie également sur le standard Web Audio Modules (WAM), qui a été conçu pour faciliter le développement de plugins audio inter-opérables (effets, instruments virtuels, claviers virtuels de piano comme contrôleurs, etc.) sortes de "VSTs pour le Web". Les DAWs sont des logiciels riches en fonctionnalités et donc particulièrement complexes à développer en termes de conception, d'implémentation, de performances et d'ergonomie. Aujourd'hui, la majorité des DAWs en ligne sont commerciaux alors que les seuls exemples open source manquent de fonctionnalités (pas de prise en charge de plugins par exemple) et ne tirent pas parti des possibilités récentes offertes (comme WebAssembly). WAM Studio a été conçu comme un démonstrateur technologique pour promouvoir les possibilités offertes par les innovations récentes proposées par le W3C. L'article met en évidence certaines des difficultés que nous avons rencontrées (par exemple, les limitations dues aux environnements sandboxés et contraints que sont les navigateurs Web, la compensation de latence quand on ne peut pas connaître le hardware utilisé, etc.). Une démonstration en ligne, ainsi qu'un *repository* GitHub pour le code source sont disponibles.

1. INTRODUCTION

La Musique Assistée par Ordinateur (MAO) est un domaine en constante évolution qui utilise des ordinateurs pour enregistrer, éditer et produire de la musique. Les stations de travail audio numérique (Digital Audio Workstations en anglais, ou "DAWs") sont des logiciels spécialement conçus pour la MAO, permettant aux utilisateurs de créer et de manipuler du contenu audio numérique ainsi que du contenu MIDI. Les plugins audio sont des modules logiciels qui ajoutent des fonctionnalités supplémentaires aux DAWs, offrant aux utilisateurs une plus grande flexibilité et un meilleur contrôle sur leur production musicale. Le marché de la MAO est né avec l'Atari ST (1985) -premier ordinateur supportant le standard MIDI- et le DAW Cubase proposée par Steinberg (1989). Peu après, le standard VST pour les plugins audio a été proposé (1997) et depuis lors,

des milliers de plugins ont été développés, pouvant être utilisés dans les principaux DAWs natifs disponibles sur le marché.

Un DAW est un logiciel riche en fonctionnalités et donc particulièrement complexe à développer en termes de conception, d'implémentation et d'ergonomie. Il permet la création de pièces multipistes en utilisant directement des échantillons audio (par exemple en incorporant dans une piste un fichier audio ou en enregistrant à partir d'un microphone ou d'une entrée de carte son), en les mélangeant, en appliquant des effets sonores à chaque piste (par exemple, réverbération, égalisation de fréquence ou auto-tune sur les voix), mais également en utilisant des pistes avec des instruments virtuels (reproduction logicielle d'un piano, d'un violon, d'un synthétiseur, d'une batterie, etc.). La piste est ensuite enregistrée au format MIDI sous forme d'événements correspondant aux notes et à des paramètres supplémentaires (comme la vitesse avec laquelle on a appuyé sur les touches d'un clavier de piano, par exemple). Ces événements permettent à la piste d'être jouée en demandant à un instrument virtuel de synthétiser le signal. Une DAW permet donc la lecture, l'enregistrement et le mixage à la fois de pistes audio et MIDI, l'édition de ces pistes (copier/couper/coller dans une piste ou entre les pistes), la gestion des effets audio en temps réel et des instruments virtuels, le mixage et l'exportation du projet final dans un format simple (par exemple, un fichier WAVE ou MP3). Dans le monde natif, ces effets et instruments sont les "plugins audio" qui étendent les capacités des DAWs standard. Depuis 1997, un marché important pour les développeurs de plugins tiers s'est développé. Quatre DAWs dominent le marché (Logic Audio, Ableton, Pro Tools, Cubase)¹ et l'existence de plusieurs formats de plugins propriétaires complique la tâche des développeurs.

D'un autre côté, le marché de la MAO sur le Web est encore émergent, les premiers DAWs en ligne sont apparus en 2008 et exploitaient la technologie Flash. Ceux utilisant HTML5 et l'API Web Audio ne sont apparus que dans la période entre 2015 et 2016 car les technologies sont beaucoup plus récentes (les premières implémentations de l'API Web Audio dans les navigateurs datent de 2012). En 2023, plusieurs DAWs basés web sont disponibles, principalement commerciaux. Un format de plugins inter-opérables appelé Web Audio Modules (WAM) existe et est soutenu par au moins un DAW en ligne.

1. <https://tinyurl.com/s4tbjzew>

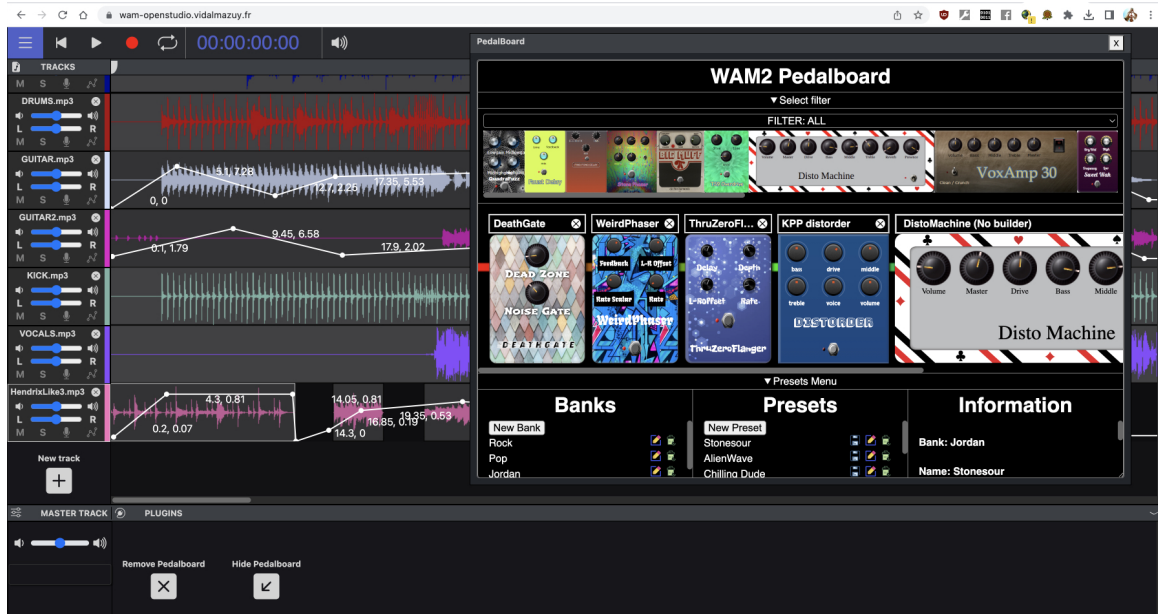


Figure 1. WAM-Studio, une station de travail audio numérique en ligne et open source. On peut voir des plugins d'effets associés à la dernière piste et des courbes d'automatisation de paramètres.

2. L'API WEB AUDIO

Depuis 2018, l'API Web Audio du W3C est une "recommandation" (une norme figée). Elle offre un ensemble de "nœuds audio" qui traitent ou produisent du son et on utilise ces nœuds depuis du code JavaScript en instanciant des classes fournies par l'API. Ces nœuds peuvent être connectés pour former un "graphe audio". Le son se déplace à travers ce graphe à la fréquence d'échantillonnage (valeur par défaut 44100Hz, cette valeur peut être modifiée) et subit des transformations [3]. Certains nœuds sont des générateurs d'ondes ou des sources sonores correspondant à une entrée de microphone ou à un fichier sonore chargé en mémoire, d'autres transforment le son. La connexion de ces nœuds dans le navigateur se fait via JavaScript et permet de concevoir une large gamme d'applications différentes impliquant le traitement audio en temps réel [6]. Les applications musicales ne sont pas les seules à nécessiter du traitement audio complexes, l'API est également conçue pour répondre à d'autres besoins, par exemple pour le développement de jeux vidéo, pour le multimédia, la visio-conférence, etc. L'API est livrée avec un ensemble limité de nœuds "standards" pour des opérations courantes telles que le contrôle de gain, le filtrage de fréquences, l'ajout de *delay*, de réverbération, pour des traitements sur la dynamique, sur la spatialisation du son 2D et 3D, etc.

En général les bibliothèques/API audio répartissent le travail entre un thread de contrôle et un thread de rendu [7, 11, 12]. L'API Web Audio ne fait pas exception et utilise également ce modèle : un thread de rendu appelé "audio thread" est chargé exclusivement du traitement du son par le graphe audio et de la livraison des échantillons sonores au système d'exploitation pour qu'ils puissent être lus par le matériel. Ce thread est soumis à des contraintes de temps

réel strictes et a une priorité élevée - s'il ne parvient pas à fournir le prochain bloc d'échantillons audio à temps (128 échantillons par défaut dans le cas de l'API Web Audio), il y aura des anomalies audibles. D'autre part, le thread de contrôle est généralement en charge d'exécuter le code JavaScript de l'interface utilisateur et d'effectuer les appels à l'API Web Audio. Il gère aussi toutes les modifications apportées au graphe audio : il permet par exemple à l'utilisateur de connecter/déconnecter des nœuds et d'ajuster leurs paramètres. Les nœuds standards sont tous des instances de sous-classes de la classe `AudioNode`. A une exception près, ils fournissent des traitements prédéfinis codés en C++ ou en Rust (Firefox) et exécutent le traitement du son dans le thread audio, mais les algorithmes utilisés ne peuvent pas être modifiés, seules les modifications de paramètres sont autorisées depuis du JavaScript. L'exception citée est l'ajout récent (2018) du nœud `AudioWorklet` qui fournit une solution pour implémenter un traitement audio personnalisé de bas niveau s'exécutant dans le thread audio [6], avec de nombreuses contraintes (pas d'opérations asynchrones, d'imports de fichiers, pas d'accès au DOM de la page HTML, etc.).

Les nœuds de l'API Web Audio peuvent être assemblés dans un graphe permettant aux développeurs de créer des effets ou instruments audio plus complexes. Voici quelques exemples construits de cette manière : effet *delay* (`DelayNode`, `BiquadFilterNode`, `GainNode`), auto-wah (`BiquadFilterNode`, `OscillatorNode`), chorus (multiples `DelayNodes` et `OscillatorNodes` pour la modulation), distorsion (`GainNode`, `WaveShaperNode`), synthétiseurs (`OscillatorNodes`), samplers (`AudioBufferNodes`), etc. Du code DSP existant dans d'autres langages tels que C/C++ ou écrit en utilisant des langages spécifiques à un domaine tels que

FAUST[13], peut être compilé en WebAssembly et exécuté dans un seul nœud AudioWorklet. Au fil des années, de nombreux effets et instruments audio de haut niveau ont ainsi été développés [4]. Cependant, il est souvent nécessaire de chaîner de tels effets et instruments audio (par exemple, le pédalier d'un guitariste est composé de pédales d'effets connectés entre elles) et lors de la composition/production musicale dans des DAWs, plusieurs effets et instruments sont en général utilisés. Ce sont des cas où les nœuds de l'API Web Audio sont de trop bas niveau, d'où la nécessité d'une unité de plus haut niveau pour représenter l'équivalent d'un plugin audio natif [1]. Pour la plate-forme Web, un tel standard de haut niveau pour les "plugins audio" et les applications "hôtes" n'existait pas avant 2015 [8]. Plusieurs initiatives ont été lancées et l'une d'entre elles est devenue un "standard communautaire" que nous détaillons dans la section suivante.

3. WEB AUDIO MODULES

En 2015, Jari Kleimola et Olivier Larkin ont proposé une norme pour des plugins Web Audio sur le Web, intitulée "Web Audio Modules" (WAM) [8]. Peu après, Jari Kleimola a participé à la création d'ampedstudio.com, l'un des premiers DAWs en ligne utilisant l'API Web Audio. En 2018, le projet WAM initial a été étendu par des chercheurs avec l'aide de développeurs de l'industrie de la MAO, ce qui a donné lieu à une norme plus polyvalente [3]. Finalement, la version 2.0 des Web Audio Modules a été publiée en 2021. Cette version vise, en plus d'établir une norme communautaire (API, SDK), à apporter plus de liberté aux développeurs (support d'outils de build, TypeScript, frameworks React, etc.), à améliorer les performances, à simplifier l'accès aux paramètres de plugin et à faciliter l'intégration dans les DAWs [5]. Nous reviendrons sur cette fonctionnalité plus tard, mais le standard WAM 2.0 utilise une conception originale pour la gestion de la communication entre les plugins et les applications hôtes qui ne repose pas sur la gestion des paramètres fournie par l'API Web Audio. La raison principale est de permettre des performances élevées dans le cas où à la fois un DAW et des plugins sont implémentés en tant qu'AudioWorklet. En effet, au moment de la conception de l'API Web Audio, les AudioWorklets n'existaient pas et certains cas d'utilisation ne pouvaient pas être pris en compte. Si le DAW est construit en utilisant des nœuds AudioWorklet pour le traitement audio, alors certaines parties du code s'exécutent dans le thread audio à haute priorité. En outre, si un plugin WAM est associé à une piste donnée dans un projet DAW, et si le plugin est lui-même construit en utilisant un nœud AudioWorklet, il dispose également de code personnalisé s'exécutant dans le thread audio. Le standard WAM a été conçu pour gérer ce cas particulier et permet une communication DAW/plugins sans quitter le thread audio. Prenons un exemple : pendant la lecture, une piste MIDI envoie des notes à un plugin d'instrument virtuel et modifie certains des paramètres de ce plugin à la fréquence d'échantillonnage (a-rate). N'oublions pas qu'un DAW peut

avoir plusieurs pistes, chacune associée à des dizaines de plugins, et que chaque plugin peut avoir des dizaines de paramètres. Le standard WAM détectera automatiquement le cas où DAW et plugins sont des AudioWorklets et optimisera en coulisse la communication (avec utilisation de mémoire partagée et de buffer tournant), sans franchir la barrière du thread audio. Pas besoin d'envoyer des événements à partir du thread de contrôle/GUI, ce qui aurait été obligatoire si la gestion des paramètres de l'API Web Audio était utilisée.

En résumé, le standard WAM simplifie la création de plugins et d'applications hôtes et permet une communication optimale entre les hôtes et les plugins.

4. ANALYSE DE L'EXISTANT : DAWs COMMERCIAUX ET OPEN SOURCE

		bandlab	ampedstudio	soundtrap	soundation	arpeggi	audiotool
Target	Desktop	X	X	X	X	X	X
	Mobile (native or responsive webapp)	X		X		X	
	Oriented to users familiar with Computer Music		X	X			
	Premium (free, with premium functionalities)	X	X	X	X		X
functionalities	Basic DAW features (recording, editing, mixing etc.)	X	X	X	X	X	X
	Audio effects and virtual instruments	X	X	X	X	X	X
	Commercials plugins support (not documented)	X	X	X	X		
	Open plugins support (WAM)		X				
	Cloud (saving project, loading, sharing, etc.)	X	X	X	X	X	X
	Collaborative (asynchronous)	X	X	X	X	X	X
	Collaborative (synchronous)			X			X
Call in app		X	X			X	
Technologies	Using AudioWorklet	X	X	?			X
	C++/wasm audio engine	X	X				
	Native mobile application	X					
	Blockchain					X	
Open Source							

Figure 2. Table de comparaison des principaux DAWs commerciaux disponibles

Les premiers DAWs en ligne basés sur l'API Web Audio sont apparus entre 2015 et 2018 et sont encore disponibles aujourd'hui : Audiotools [9], Bandlab, Aamped-Studio, Soundtrap [10], Soundation. Ces DAWs ont en commun le fait qu'ils sont commerciaux, fermés et ont fait l'objet de très peu de publications académiques. Ils facilitent la collaboration à distance sur des projets musicaux, notons que la pandémie de COVID-19 (2021-2022) leur a été profitable et a augmenté leur popularité. Le mode de collaboration varie (synchrone comme Google Docs ou asynchrone grâce au partage de liens), ainsi que les outils de communication fournis (chat, vidéo-conférence), mais

tous ces DAWs proposent les fonctionnalités classiques : enregistrement audio et MIDI, édition de pistes, mixage, support pour effets et instruments virtuels, etc. Certains ont été conçus principalement pour les ordinateurs de bureau, tandis que d'autres sont particulièrement adaptés aux appareils mobiles.

Ils diffèrent principalement en termes d'ergonomie : certaines de ces applications sont destinées à un public très large et ont mis l'accent sur la simplicité (BandLab, SoundTrap), tandis que d'autres ont choisi un aspect plus "professionnel" et sobre, comme AmpedStudio. De son côté, Audiotools est davantage un "studio virtuel" qu'un DAW pur et trouve son marché principalement dans le domaine de l'éducation. Les détails de mise en œuvre de ces applications sont inconnus car le code source n'est pas accessible au public. Cependant, à travers des publications académiques limitées et des présentations/interviews, on sait qu'AmpedStudio utilise un moteur C++ cross-compilé en WebAssembly et des AudioWorklets, tandis que Soundtrap a principalement utilisé les nœuds standard de l'API Web Audio. BandLab est censé avoir un noyau écrit en C++ et s'appuie également sur WebAssembly/AudioWorklet, avec peut-être une version mobile spécifique. Audiotools fonctionne également dans des nœuds AudioWorklet avec un portage du moteur audio initialement écrit en Flash. Récemment, un nouveau DAW appelé Arpeggi.io est apparu, mettant en avant l'utilisation de la technologie blockchain (pour suivre qui, quand et comment les sons et la musique sont utilisés) et des NFTs pour monétiser les créations.

Ces DAW diffèrent également dans la manière dont ils gèrent les effets et les instruments audio. Il est évident que certains prennent en charge des plugins externes : AmpedStudio prend en charge la norme WAM -les développeurs de la société ont contribué à sa création- et le site web a une boutique pour "activer" des plugins premium. Les autres DAWs en ligne commerciaux semblent utiliser un format propriétaire, tout en intégrant des portages tiers de plugins natifs (SoundTrap propose le célèbre plugin "Auto-tune" d'Antares). Propellerheads, une entreprise bien connue pour son DAW natif Reason, a également publié des versions web de son synthétiseur Europa sous forme de plugin Web (disponible maintenant dans AmpedStudio et dans Soundation, il s'agit d'un portage de Plugin Rack, un standard de plugins propre à Propellerheads dans le monde natif).

Jusqu'à présent, AmpedStudio est le seul DAW à prendre en charge une norme de plugins ouverte : Web Audio Modules. La figure 2 illustre les similitudes et les différences entre les DAW en ligne commerciaux cités.

Dans le monde de l'open source, le choix est limité : GridSound est un DAW développé depuis 2015 ² qui prend en charge l'audio et le MIDI. Bien que les effets et instruments proposés demeurent simples, GridSound est un DAW entièrement fonctionnel. Il ne prend pas en charge de plugins.

2. <https://gridsound.com/>. Voir également la présentation vidéo de GridSound à la conférence ADC 2021 : <https://www.youtube.com/watch?v=eJTtENwRxnA>

Il existe également des bibliothèques JavaScript populaires pour développer un lecteur/enregistreur multipiste, telles que wavesurfer.js ³, peaks.js ⁴ ou waveform-playlist ⁵, et des éditeurs de fichiers audio de type audacity tels que AudioMass ⁶.

Aucune de ces initiatives open source n'utilise de traitement DSP personnalisé de bas niveau (pas d'AudioWorklets), ni ne se soucie d'optimiser la communication entre DAW/plugin, ni n'utilise de plugins externes. C'est la raison principale pour laquelle nous avons développé WAM Studio : en tant que démonstrateur de ces techniques.

5. WAM STUDIO DESIGN ET IMPLÉMENTATION

Wam-Studio est un outil en ligne pour créer des projets audio que l'on peut imaginer comme de la musique multipistes. Chaque piste correspond à une "couche" différente de contenu qui peut être enregistré, édité, joué ou simplement intégré (en utilisant des fichiers audio, par exemple). Certaines pistes peuvent être utilisées pour contrôler des instruments virtuels : dans ce cas, elles ne contiennent que les notes qui doivent être jouées, avec quelques métadonnées. Des pistes peuvent être ajoutées ou supprimées, jouées isolément ou avec d'autres pistes. Elles peuvent également être "armées" pour l'enregistrement, et lorsque l'enregistrement commence, toutes les autres pistes joueront simultanément, tandis que les pistes armées enregistreront un nouveau contenu.

5.1. Les pistes

Dans un DAW, chaque piste est un conteneur de données liées à l'audio qui s'accompagne d'une représentation interactive de ces données, de fonctionnalités d'édition et de traitement ainsi que de quelques paramètres par défaut tels que le volume et le panoramique gauche/droite de la piste.

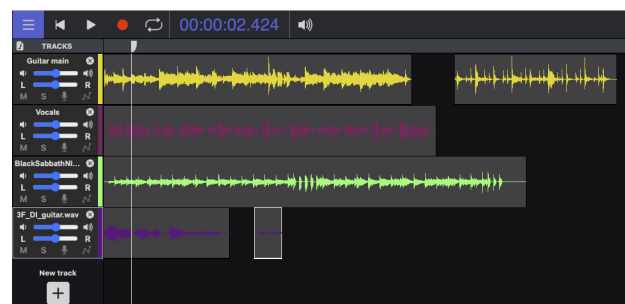


Figure 3. Pistes et régions dans la GUI du DAW Wam-Studio. Les régions peuvent être déplacées, supprimées, etc.

La figure 3 montre des pistes audio dans WAM-Studio avec l'affichage de la forme d'onde des buffers audio associés et les contrôles/paramètres par défaut des pistes,

3. <https://wavesurfer-js.org/>

4. <https://github.com/bbc/peaks.js/>

5. <https://github.com/naomiario/waveform-playlist>

6. <https://audiomass.co/>

sur le côté gauche (mute/solo, armement pour l'enregistrement, volume, panoramique stéréo, affichage des courbes d'automatisation). Comme de nombreuses pistes peuvent être affichées, scrollées pendant la lecture, zoomées et éditées, nous avons utilisé la bibliothèque pixi.js pour gérer de manière efficace le dessin et les interactions dans un canvas HTML. Cette bibliothèque utilise un rendu WebGL accéléré par GPU et offre de nombreuses fonctionnalités pour la gestion de plusieurs calques sur un seul canvas HTML5 (Figure 3). On voit sur la Figure 5 que chaque piste peut également être associée à un ensemble de plugins pour ajouter des effets audio ou générer de la musique (dans le cas de plugins instrumentaux).

La Figure 4 montre le graphe audio correspondant à la chaîne de traitement d'une piste audio. Le son va de gauche à droite : d'abord le "lecteur/enregistreur/éditeur de piste" est implémenté en tant que nœud AudioWorklet, en utilisant du code personnalisé pour le rendu d'un buffer audio, puis le signal de sortie a son gain et son panoramique stéréo ajustés, ensuite nous avons un autre nœud AudioWorklet pour le rendu du volume dans un canvas (VU-mètre), et enfin nous trouvons une chaîne de plugins WAM pour ajouter des effets audio.

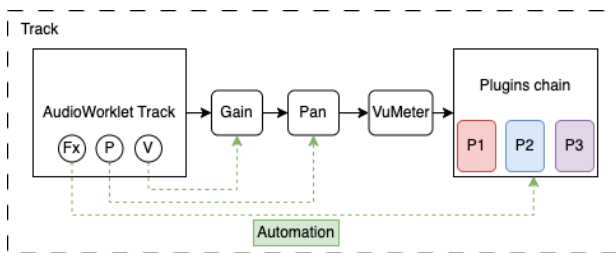


Figure 4. Graphe d'une piste audio dans Wam-Studio

Il existe deux types de pistes : **les pistes "audio"** : elles contiennent de l'audio enregistré, tel qu'une prise vocale, un enregistrement de guitare ou tout autre type de signal sonore, qui sont généralement rendus graphiquement avec une forme d'onde représentant le signal stocké (sous forme d'un ensemble d'échantillons sonores). Ces pistes peuvent être éditées, traitées et mélangées en copiant/coupant et collant des échantillons dans le buffer audio associé à la piste. Le signal de sortie des pistes audio peut être traité par une chaîne de plugins d'effet audio. **Les pistes "MIDI"** : elles contiennent des données MIDI (le pitch - la note qu'un instrument jouerait, sa vélocité et sa durée). Les données MIDI ne contiennent pas d'informations audio mais sont plutôt utilisées pour contrôler des instruments virtuels tels que des synthétiseurs, des samplers, des boîtes à rythmes, etc. Les pistes MIDI peuvent être éditées, auquel cas il est possible de modifier les événements MIDI stockés dans la piste dans un affichage de type matriciel. Les pistes MIDI sont généralement associées à un ou plusieurs plugins WAM d'instruments virtuels. Les événements MIDI peuvent cibler tous les plugins d'instruments de la piste ou certains en particulier.

Les pistes sont généralement organisées verticalement dans l'interface utilisateur et peuvent être gérées séparément

en termes de volume, de panning stéréo, de plugins et d'automatisation des paramètres de ces plugins d'effets ou d'instruments (comme le montre la Figure 5).



Figure 5. Une chaîne de plugins associés à la piste sélectionnée.

Chaque sortie de piste est connectée à une "piste maître" (master track) où il est possible d'ajuster globalement le volume et le panning stéréo du mix final. Comme pour toutes les autres pistes, il est également possible d'associer une chaîne d'effets audio à la piste maître (par exemple pour ajuster la dynamique et les fréquences finales). Tous les effets audio et instruments virtuels sont des plugins WAM dans le cas de notre DAW, et tous leurs paramètres sont automatisables⁷. Cette conception offre un degré de contrôle et d'adaptabilité étendu et permet aux utilisateurs de mélanger et de manipuler le son de chaque piste avec une grande précision et sophistication, rendant ainsi plus facile la création de productions audio complexes, le tout sans quitter le navigateur Web. WAM-studio est le seul logiciel open source à proposer l'ensemble de ces fonctionnalités.

Une piste DAW est également capable de "rendre" la piste en un signal audio audible (lorsque la piste est en lecture). Lorsque l'on appuie sur le bouton de lecture du DAW, toutes les pistes sont rendues simultanément, et le signal de sortie final est ce que l'on appelle "le mix". Les DAWs disposent également d'une option de "rendu hors ligne" pour exporter le mix final sous forme de fichier, sur le disque dur local ou sur le cloud (en utilisant l'OfflineAudioContext⁸ fourni par l'API Web Audio).

5.2. Le coeur des pistes est un AudioWorklet Processor

Concentrons-nous pour le moment sur une piste de type audio. L'API Web Audio fournit le nœud AudioBufferSourceNode qui est un conteneur pour un buffer audio, et qui permet la lecture d'un buffer audio stocké en mémoire. Cette approche simple à mettre en oeuvre est néanmoins non adaptée pour un DAW devant gérer

7. A la fréquence d'échantillonnage (a-rate), du quantum audio (k-rate), personnalisée : cela dépend de l'implémentation du plugin au format WAM

8. Contrairement à un AudioContext standard, qui sert à créer les nœuds du graphe audio pour un rendu "live", un OfflineAudioContext sert lui, à générer aussi rapidement que possible un graphe audio et à envoyer le résultat dans un AudioBuffer, qui peut ensuite être converti en format WAVE par exemple.

l'automation de paramètres. Cette tâche consiste à interpoler, pendant la lecture ou l'enregistrement multipiste, les paramètres des plugins et des pistes à de hautes fréquences (à la fréquence d'échantillonnage *a-rate* ou de contrôle *k-rate*). Un paramètre automatisé peut être celui de n'importe quel plugin associé à n'importe quelle piste. Il peut y avoir potentiellement des dizaines, des centaines de paramètres à automatiser. Cela représente une quantité substantielle d'informations qui peut être échangée entre le DAW et les plugins WAM, ce qui fait de la performance un aspect crucial qui nécessite une attention particulière. En concevant notre propre lecteur audio de bas niveau (au lieu d'utiliser le nœud standard `AudioBufferSourceNode`), nous avons un contrôle total sur le comportement de lecture/enregistrement de chaque piste. Nous avons ainsi conçu le noyau de chaque piste audio comme un "AudioWorklet processor", qui permet l'exécution de code DSP personnalisé dans le thread audio à très haute priorité. Plus précisément, ce noyau est une instance d'une sous-classe de la classe `WamProcessor` fournie par le SDK des Web Audio Modules, qui hérite de la classe `AudioWorkletProcessor`. Selon l'API, chaque processeur implémente une méthode `process(input, output)` qui sera appelée à la fréquence d'échantillonnage. À l'intérieur de cette méthode, nous traitons les échantillons sonores et procédons au rendu du buffer audio, mais nous pouvons également envoyer des données aux plugins WAM. La classe `WAMProcessor` dont nous héritons fournit en effet des méthodes pour gérer efficacement la communication entre le DAW et les plugins.

Avec l'API Web Audio, les `AudioWorklets` sont composés de deux composants, reflétant la nature multi-threadée de l'environnement : le nœud `AudioWorkletNode` et le `AudioWorkletProcessor`. Le standard WAM les étend avec les classes `WamNode` et `WamProcessor`, qui s'exécutent respectivement dans le main thread du navigateur (thread de la GUI) et dans le thread audio. WAM-Studio est une application hôte WAM, et ses pistes sont donc mises en œuvre en tant que sous-classes de `WamProcessor`. Les plugins WAM associés à ces pistes peuvent également avoir leur cœur DSP implémenté en tant que sous-classes de `WamNode` avec un `WamProcessor`, mais ce n'est pas obligatoire car certaines implémentations de plugins WAM utilisent plutôt un graphe composé d'une combinaison de nœuds, dans ce cas ils étendent la classe `WAM CompositeNode`. Quelle que soit l'implémentation des plugins le code de communication de l'hôte vers le plugin est le même, et les optimisations mise en œuvre se font en coulisse (dans le WAM SDK).

Cette abstraction permet une inter-opérabilité transparente entre les hôtes et les plugins WAM, indépendamment de la mise en œuvre sous-jacente et sur tous les threads. En d'autres termes : si une piste est un `WamProcessor` et que les plugins sont également des `WamProcessor`, leur code DSP s'exécute dans le thread audio et une communication hautement optimisée peut être réalisée : les `Shared Array Buffers` peuvent être utilisés pour la communication entre DAW et plugins. Si un plugin n'est pas un `WamProcessor`, alors le code DAW qui "communique" avec lui reste

inchangé et la mise en œuvre sous-jacente sera moins optimale et impliquera le franchissement de la barrière des threads. Pour gérer plusieurs plugins en chaîne, le standard WAM utilise un objet singleton, appelé `WamEnv`, qui est attaché à la portée globale du thread audio et sert de médiateur pour les interactions entre les hôtes et les plugins. Les plugins sont structurés en `WamGroup`, regroupés dans un "WamEnv", qui encapsule le code de la version du SDK WAM utilisé. Si une nouvelle version du SDK sort, il sera ainsi possible de charger les plugins dans un environnement `WamEnv` protégé, chaque plugin utilisant sa propre version du SDK. Chaque groupe de plugins (`WamGroup`) comprend les plugins créés par un hôte spécifique et il est possible d'envoyer des événements (MIDI par exemple) à tous les plugins à la chaîne. Un effort considérable a été déployé pour réduire le nombre d'hypothèses faites par l'API WAM concernant la mise en œuvre de l'hôte, avec `WamEnv` et les instances de `WamGroup` étant les seuls objets du système WAM qui sont fournis par l'hôte. Le `WamEnv` est alloué lors de la création du DAW et les instances de `WamGroup` sont associées à chaque piste.

5.3. Gestion des chaînes de plugins

Les chaînes de plugins sont gérées à l'aide d'un plugin WAM spécial qui agit également en tant que "mini-hôte" (Figure 7). Nous l'avons appelé le WAM pedalboard [1]. Il se connecte à un serveur de plugins qui renvoie la liste des plugins disponibles sous forme de tableau JSON d'URIs (un plugin WAM peut être chargé simplement à l'aide d'une importation dynamique et de son URI, voir [5]). À partir de cette liste d'URIs, les descripteurs de plugins WAM sont récupérés, qui contiennent des méta-données sur les plugins : nom, version, fournisseur, URI d'image miniature, etc. Lorsque le plugin pedalboard est affiché dans le DAW, la chaîne de plugins actifs est vide, et les plugins peuvent être ajoutés à la chaîne de traitement, supprimés, ré-ordonnés et leurs paramètres peuvent être définis.



Figure 6. Le plugin WAM Pedalboard qui gère les chaînes de plugins associés à chaque piste. Ici chargé dans une simple page HTML hôte, en mode stand-alone.

Toute configuration peut être enregistrée en tant que preset nommé (par exemple, "son de guitare crunch 1"). Les presets peuvent être organisés en banks ("rock", "funk", "blues"). La gestion de l'organisation et de la nomination des banks et des presets relève de la responsabilité du pedalboard plugin. Les paramètres exposés par ce plugin correspondent à l'ensemble des paramètres du preset ac-

tif (c'est-à-dire la somme des paramètres des plugins du preset), et peuvent être automatisés par le DAW. L'API Web Audio Modules permet d'envoyer des événements au DAW lorsque des modifications se produisent dans la configuration du plugin (ajout ou suppression), et un menu fourni dans le DAW avec chaque piste pour sélectionner les paramètres pouvant être automatisés est automatiquement mis à jour (Figure 7).



Figure 7. Chaque piste a un menu d'automatisation qui se met à jour en fonction des plugins associés.

Tous les plugins WAMs implémentent également des méthodes getState/setState pour sérialiser/dé-sérialiser leur état. Lorsqu'un projet est enregistré, l'état de chaque piste, des buffers audio, etc. est enregistré, ainsi que l'état de la configuration de chaque plugin.

5.4. Enregistrement, gestion de la latence

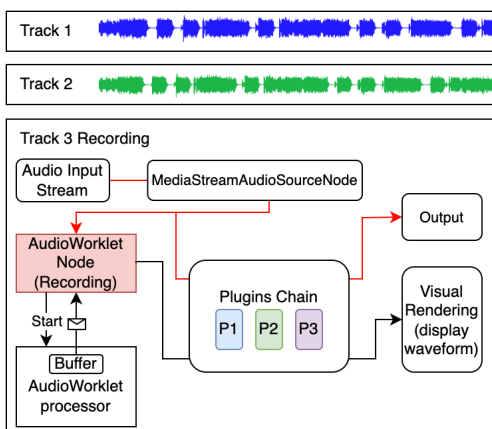


Figure 8. Schéma logique de l'enregistrement audio dans une piste de WAM-Studio

La figure 8 montre le graphe audio correspondant au processus d'enregistrement d'une piste : nous devons d'abord obtenir un flux multimédia à partir du microphone de l'utilisateur ou d'une entrée de carte son à l'aide de l'API MediaDevices. Ensuite, nous pouvons utiliser un MediaStreamSourceNode de l'API Web Audio dans le graphe audio, en passant le flux multimédia à son constructeur. C'est comme cela que nous exposons un flux audio live à l'API Web Audio. Dans un deuxième temps, nous avons besoin d'un moyen d'enregistrer ce flux dans un buffer. Cela peut être fait en utilisant l'API W3C MediaRecorder

ou en utilisant des solutions de bas niveau à l'aide d'un AudioWorklet. La solution MediaRecorder a pour avantage sa simplicité, mais tous les tests que nous avons effectués ont montré que c'est une solution peu fiable car le temps nécessaire pour allouer des buffers audio et commencer l'enregistrement est imprévisible, même lorsque nous effectuons des mesures/calibrations de latence, comme expliqué plus loin. Nous avons finalement opté pour une solution de bas niveau plus précise qui utilise un WamProcessor dont le rôle est d'enregistrer en continu les échantillons sonores dans un ring buffer contenant à peu près une seconde de son numérisé, situé dans une mémoire partagée avec un Worker JavaScript qui lui tourne dans un thread moins prioritaire et dont le rôle est de consommer les échantillons envoyés par le WamProcessor, et les copier à la fin d'un buffer qui grossit au fur et à mesure. Cette approche évite des allocations dans le thread audio, elle est également plus robuste contre un éventuel stress CPU. Le Worker envoie également sous forme de message le buffer audio courant à un code JavaScript exécuté dans le thread de la GUI. Ce dernier peut ainsi dessiner la forme d'onde du signal audio enregistré qui grandit visuellement au fur et à mesure que l'enregistrement avance. Lors qu'on stoppe l'enregistrement, la piste est prête à être jouée puisque le buffer audio a été mis à jour au fur et à mesure.

Compensation de latence : Le processus d'enregistrement avec un DAW est plus contraint et structuré que dans un simple enregistreur de mémos audio, car nous devons prendre en compte différents types de latences. La latence d'entrée est le temps entre la capture d'un signal audio par le dispositif d'entrée (carte son) et son traitement par le contexte audio. Elle dépend du système d'exploitation, de la configuration du dispositif d'entrée, du temps de traitement du système audio et de la taille du buffer utilisé par le contexte audio. La latence de sortie représente le temps entre la génération d'un son et sa perception par l'utilisateur. La somme de ces deux latences représente la latence round-trip et peut être mesurée à l'aide d'un dispositif d'enregistrement externe avec deux microphones : un sur la source du son physique (par exemple, sur le corps d'une guitare branchée sur une carte son), et un devant les haut-parleurs. Frappez le corps de la guitare, enregistrez la sortie et comparez les signaux d'entrée et de sortie. Nous avons effectué de telles mesures dans le passé[2]. On peut aussi l'estimer avec un programme émettant des sons (en général des sons de type métronome) et en enregistrant ces mêmes sons en plaçant un micro devant les haut-parleurs. Paul Adenot propose une implémentation d'un outil permettant de mesurer cette latence et aussi la latence de sortie ⁹. Le résultat est en général moins précis que lors de la mesure avec des outils externes.

Par ailleurs, l'API Web Audio expose une propriété outputLatency de l'AudioContext dont la valeur est une approximation de la latence de sortie, soit le nombre de secondes entre l'audio atteignant l'AudioDestinationNode

9. Voir le billet de blog de Paul Adenot <https://blog.paul.cx/post/audio-video-synchronization-with-the-web-audio-api/>

(conceptuellement, la sortie du graphe de traitement audio) et le dispositif de sortie audio. Si on sait mesurer la latence round-trip, alors on peut connaître la latence d'entrée puisque latence d'entrée + latence de sortie = latence round-trip.

Lors de l'enregistrement d'une piste de guitare, par exemple, d'autres pistes telles que des pistes de batterie et de basse peuvent être jouées simultanément. Le son de la guitare est traité en temps réel par une chaîne de plugins (simulation d'amplificateur de guitare, égaliseur, retard, etc.) et doit être entendu en temps réel par le guitariste, en synchronisation avec les autres pistes et sans latence perceptible lorsqu'il est joué (et enregistré). C'est pourquoi la latence round-trip globale doit être aussi faible que possible. Nous avons représenté la route permettant de s'entendre jouer pendant l'enregistrement avec le chemin rouge dans la Figure 8, une partie du signal traverse la chaîne de plugins et peut être entendue (*wet route*) et le signal non traité est enregistré (*dry route*, vers l'AudioWorklet). En utilisant une taille de buffer de 128 échantillons (valeur par défaut lors de la création d'un AudioContext), la latence round-trip mesurée sur MacOS est de 23 ms avec le navigateur Chrome, 15ms avec Firefox, suffisamment confortable et comparable à ce que nous pouvons obtenir avec des applications natives dans des cas d'utilisation similaires (guitare, Logic Audio, plugin de simulation d'amplificateur de guitare, taille de buffer audio de 128), même avec des guitaristes jouant très vite[2].

Néanmoins, lors de la lecture simultanée de la nouvelle piste enregistrée, un décalage de plusieurs millisecondes est visible et audible, la piste nouvellement enregistrée est "en retard" par rapport aux autres. Il est nécessaire de décaler en arrière dans le temps la nouvelle piste. Cette opération s'appelle la "compensation de latence", et la valeur de compensation est égale à *-valeur de la latence d'entrée*. Nous avons vu que cette valeur peut être calculée si on connaît la latence round-trip (puis que la latence de sortie est fournie par l'AudioContext).

Dans WAM Studio, un menu de configuration permet de calculer automatiquement la compensation de latence en approchant un micro des haut parleurs et en lançant le processus de calibration (le DAW émet des sons, enregistre le résultat et compare les temps d'émission et de réception). Pour une configuration matérielle donnée cette opération n'est à effectuer qu'une seule fois.

Le DAW AmpedStudio propose une approche similaire. Les auteurs de SoundTrap ont expliqué qu'ils utilisent plutôt des tables prédéfinies avec des entrées pour les paires OS/carte son les plus courantes, dont la latence d'entrée a été mesurée à la main et codées en dur dans le code[10]. SoundTrap utilise ensuite des heuristiques pour déterminer la configuration (vérification de l'OS, deviner le modèle de la carte son en regardant le nombre d'entrées/sorties exposées en utilisant l'API MediaDevices, etc.).

D'autres optimisations classiques ont été mises en place dans WAM-studio : dès que les pistes sont armées pour l'enregistrement, on alloue le buffer tournant, on démarre les Workers (tâches de fond JavaScript) d'enregistrement,

on pré-alloue les buffers audio et on ne commence à enregistrer que lors de l'appui sur les boutons "record" puis "play", à ce moment là, l'enregistrement à proprement parler commence. Ceci évite de perdre du temps au démarrage, avec les allocations et le lancement des Workers.

6. CONCLUSION

À ce jour, WAM-Studio est un exemple de DAW open source utilisant des fonctionnalités avancées telles que la prise en charge de plugins externes, des communications DAW/plugins à haute performance, la lecture, l'enregistrement et l'édition de pistes audio avec compensation de latence. En tant que démonstrateur, il montre les capacités de l'API Web Audio et comment le standard Web Audio Modules peut être utilisé pour faciliter le développement d'applications audio web complexes. La prise en charge des pistes MIDI est prévue à court terme ¹⁰ et le développement est actif. Le code source du DAW peut être trouvé sur GitHub ¹¹, et l'application est également disponible en ligne ¹².

7. REFERENCES

- [1] Michel Buffa, Pierre Kouyoumdjian, Quentin Beauchet, Yann Forner, and Michael Marynowic. "Making a guitar rack plugin -WebAudio Modules 2.0", *In Web Audio Conference 2022.*, Cannes, France, 2022. <https://hal.inria.fr/hal-03812948>.
- [2] Michel Buffa and Jerome Lebrun. 2017. "Real time tube guitar amplifier simulation using WebAudio" *In Proceedings of the Web Audio Conference (Queen Mary Research Online (QMRO) repository)*. <http://qmro.qmul.ac.uk/xmlui/handle/123456789/26089>. Queen Mary University of London, London, United Kingdom. <https://hal.univ-cotedazur.fr/hal-01589229>
- [3] Michel Buffa, Jerome Lebrun, Jari Kleimola, Oliver Larkin, and Stephane Letz. 2018. "Towards an open Web Audio plugin standard" *In WWW2018 - The WebConf 2018 : The Web Conference, 27th International World Wide Web Conference*. Lyon, France. <https://doi.org/10.1145/3184558.3188737>
- [4] Michel Buffa, Jerome Lebrun, Shihong Ren, Stéphane Letz, Yann Orlarey, Romain Michon, and Dominique Fober. 2020. "Emerging W3C APIs opened up commercial opportunities for computer music applications" *In The Web Conference 2020 DevTrack*. Taipei, Taiwan. <https://doi.org/10.13140/RG.2.2.16456.19202>
- [5] Michel Buffa, Shihong Ren, Owen Campbell, Jari Kleimola, Oliver Larkin, and Tom Burns. 2022. "Web Audio Modules 2.0 : An Open Web Audio Plugin

10 . Nous avons écrit des démonstrations de lecture de pistes MIDI à l'aide de WAMs sur le site de tutoriels WAM : <https://wam-examples.vidalmazuy.fr/>

11 . <https://github.com/Brotherta/wam-refont>

12 . <https://wam-openstudio.vidalmazuy.fr/>

Standard” In *WWW '22 : The ACM Web Conference 2022*. ACM, Virtual Event, France, 364–369. <https://doi.org/10.1145/3487553.3524225>

- [6] Hongchan Choi. 2018. ”Audioworklet : the Future of Web Audio” In *Proceedings of the International Computer Music Conference*. Daegu, South Korea, 110–116.
- [7] François Déchelle, Riccardo Borghesi, Maurizio De Cecco, Enzo Maggi, Butch Rován, and Norbert Schnell. ”jMax : an environment for real-time musical applications” In *Computer Music Journal* 23, 3, 50–58. 1999.
- [8] Jari Kleimola and Oliver Larkin. ”Web Audio Modules” In *Proceedings of the Sound and Music Computing Conference 2015*.
- [9] Bojan Lazarevic and Danijela Scepanovic. ”Unrevealed Potential in Delivering Distance Courses : the Instructional Value of Audio” In *eLearning and Software for Education* (2010).
- [10] Fredrik Lind and Andrew 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.
- [11] James McCartney. ”Rethinking the computer music language : Super collider” In *Computer Music Journal* 26, 4, 61–68. 2002.
- [12] Miller Puckette. ”FTS : A real-time monitor for multi-processor music synthesis” In *Computer music journal* 15, 3, 58–67. 2002.
- [13] Shihong Ren, Stéphane Letz, Yann Orlarey, Romain Michon, Dominique Foer, et al. ”Using Faust DSL to Develop Custom, Sample Accurate DSP Code and Audio Plugins for the Web Browser”. *Journal of the Audio Engineering Society*, 68 (10), pp.703-716. 2020.