



Applications audio web avec Faust : innovations techniques au service de la transmission

Web audio applications with Faust : technical innovations for transmission

Stéphane Letz¹, Jeanne Ribeiro²

¹Responsable de la recherche, GRAME CNCM letz@grame.fr

²Responsable du développement artistique et culturel des territoires, GRAME CNCM
ribeau@grame.fr

Résumé

Cet article présente les développements récents de l'écosystème Faust pour le Web, avec un focus sur la modernisation des outils et infrastructures, notamment à travers le projet faustwasm, qui s'appuie sur les technologies WebAssembly et les langages TypeScript/JavaScript pour simplifier la création d'applications musicales interactives, performantes et portables. L'utilisation du modèle des Progressive Web Applications (PWA) est détaillée, permettant de contourner les limitations des plateformes propriétaires, tout en offrant des solutions plus légères et multiplateformes plus faciles à distribuer. Le portage de projets emblématiques comme SmartFaust ou GameLan utilise alors ces avancées technologiques, facilitant leur accessibilité pour des concerts participatifs et des ateliers pédagogiques. En particulier, ces applications transforment les smartphones en instruments de musique, exploitant les capteurs de mouvement pour proposer des interactions gestuelles intuitives et immédiates. Leur simplicité d'utilisation, combinée à l'absence de prérequis techniques, en fait des outils particulièrement adaptés à l'éducation artistique et à la sensibilisation aux musiques électroacoustiques. L'article expose les bénéfices pédagogiques observés : développement de l'écoute, travail sur la coordination et la mémoire gestuelle, stimulation de la créativité individuelle et collective, etc. Enfin, des exemples concrets d'intégration de ces outils dans des contextes scolaires, sociaux ou culturels seront présentés, ainsi que les perspectives offertes par cette approche, sans occulter certaines des contraintes matérielles ou organisationnelles qui restent toujours à maîtriser.

Mots-clefs : Faust ; WebAssembly ; Applications web progressives (PWA) ; Musique interactive ; Pédagogie musicale ; Capteurs de mouvement

Abstract

This article presents recent developments in the Faust ecosystem for the web, with a focus on the modernisation of tools and infrastructure, notably through the faustwasm project, which uses WebAssembly technologies and TypeScript/JavaScript languages to simplify the creation of interactive, high-performance, portable music applications. The use of the Progressive Web Applications (PWA) model is detailed, allowing the limitations of proprietary



platforms to be circumvented, while offering lighter, multi-platform solutions that are easier to distribute. The porting of flagship projects such as SmartFaust and GameLan makes use of these technological advances, facilitating their accessibility for participatory concerts and educational workshops. In particular, these applications transform smartphones into musical instruments, using motion sensors to offer intuitive and immediate gestural interactions. Their ease of use, combined with the absence of technical prerequisites, makes them particularly suitable tools for arts education and raising awareness of electroacoustic music. The article outlines the educational benefits observed: development of listening skills, work on coordination and motor memory, stimulation of individual and collective creativity, etc. Finally, concrete examples of how these tools can be integrated into educational, social and cultural contexts will be presented, along with the prospects offered by this approach, without overlooking some of the material and organisational constraints that still need to be overcome.

Keywords : Faust ; WebAssembly; Progressive Web Applications (PWA) ; Interactive music; Music education; Motion sensors

1. Introduction

Depuis plus d'une décennie, le langage FAUST (Functional Audio Stream) s'est imposé comme un outil facilitant le développement d'applications musicales interactives, en particulier dans le cadre de projets exploitant les capacités gestuelles des smartphones. Initialement déployées sous forme d'applications natives sur iOS et Android, des réalisations comme SmartFaust et GameLan ont permis d'explorer des approches innovantes en matière d'interaction musicale participative et de transmission pédagogique.

Toutefois, la maintenance parallèle de deux bases de code spécifiques à chaque plateforme, couplée aux exigences des systèmes de distribution propriétaires (App Store et Google Play Store), a progressivement révélé ses limites : lourdeur de gestion, dépendance aux critères évolutifs de validation, obsolescence technique. Parallèlement, l'évolution rapide du Web, notamment l'émergence de technologies comme WebAssembly, la Web Audio API et les Progressive Web Applications (PWA), des applications Web modernes offrant une expérience utilisateur comparable à celle des applications natives, a ouvert de nouvelles perspectives.

Dans ce contexte, une refonte complète a été entreprise, visant à recentrer le développement sur une architecture Web unique, plus pérenne et plus accessible. Le projet *faustwasm*, développé autour de la compilation vers WebAssembly et accompagné d'enrobages écrits en TypeScript/JavaScript, constitue une étape centrale de cette transition. Ce choix technique permet non seulement d'assurer des performances audio temps réel très proches des applications natives, mais aussi de simplifier considérablement la distribution et l'accès aux applications.

Au-delà de l'amélioration technique, ce basculement vers le Web offre également un cadre particulièrement favorable à la démocratisation des pratiques musicales numériques. L'utilisation de PWA facilite l'intégration de ces outils dans des contextes pédagogiques variés : ateliers scolaires, projets participatifs, médiation culturelle, etc.



Cet article présente en détail cette évolution, du développement initial des applications natives jusqu'au portage vers le Web, en soulignant à la fois les bénéfices observés dans les pratiques artistiques et pédagogiques, ainsi que les défis techniques et organisationnels rencontrés. Le portage des projets historiques, tels que SmartFaust ou GameLan vers ces environnements plus flexibles et modernes, permet de démocratiser la transmission musicale et l'interaction gestuelle.

2. État de l'art

L'usage des smartphones et des technologies Web pour la création musicale interactive a donné lieu à de nombreuses expérimentations depuis plus d'une décennie dans un cadre artistique ou pédagogique. Cette section présente quelques approches, avec des solutions natives spécifiques aux plateformes mobiles et d'autres exploitant directement les technologies Web.

2.1. Applications mobiles natives pour la création musicale

Avec l'essor des smartphones équipés de capteurs de mouvement, plusieurs projets ont exploré leur potentiel pour la création musicale gestuelle.

Parmi les plus emblématiques, on retrouve CoMo — Collective Movements, développé par l'équipe ISMM de l'IRCAM¹. Cette suite d'applications exploite les accéléromètres et gyroscopes pour proposer une variété d'usages : performances artistiques (CoMo-Elements), enseignement (CoMo-Education), rééducation (CoMo-Rééducation) ou encore direction chorale (CoMo-Vox). Les applications reposent sur une architecture client-serveur centralisée, nécessitant une connexion WiFi et une infrastructure réseau dédiée, ce qui peut constituer une contrainte logistique.

SoundWorks également développé à l'IRCAM est une plateforme JavaScript destinée à la création d'applications audio interactives en réseau. Elle permet de synchroniser plusieurs appareils (smartphones, ordinateurs) pour des expériences musicales collaboratives ou performatives. En se connectant simplement à une page web via leur navigateur, les utilisateurs peuvent générer des sons et des effets lumineux en temps réel grâce aux capteurs intégrés de leurs téléphones, tels que l'accéléromètre et le gyroscope.

Le projet WAXML de Hans Lindetorp et Kjetil Falkenberg de KTH/Stockholm montre un large éventail d'applications WebAudio sur smartphone développées à des fins éducatives². Des démonstrations très impressionnantes car couplées avec de nombreuses applications enrichissent l'utilisation des simples capteurs gyroscopiques. En particulier l'utilisation de Media Pipe pour la reconnaissance des expressions du visage, des gestes, du squelette, etc.

Sur l'Apple Store d'Apple, on peut trouver des applications comme GyroSynth, qui transforme l'iPhone en un instrument musical inspiré du thérémine, en exploitant l'orientation du téléphone pour contrôler la hauteur des sons³.

¹ <https://apps.ismm.ircam.fr/como>

² <https://waxml.org>

³ <https://apps.apple.com/fr/app/gyrosynth/id386527164>



Ces projets illustrent la diversité des usages artistiques des smartphones dans une logique de détournement technologique, mais ils restent parfois contraints par la nécessité de développer et maintenir des applications spécifiques pour chaque système d'exploitation.

2.2. Le Web comme plateforme musicale

Grâce à l'introduction de la Web Audio API et de standards comme WebAssembly, la plateforme Web est progressivement apparue comme un environnement viable pour la synthèse et traitement sonore en temps réel couplé avec des interfaces musicales,

Dès 2013, Roberts *et. al.* soulignaient les avantages du navigateur comme plateforme musicale : portabilité, facilité d'accès, absence de compilation locale. Leur travail a ouvert la voie à l'émergence de nombreux environnements exploitant ces possibilités.

Le projet WebChucK, constitue un exemple récent d'environnement de programmation musicale embarqué dans le navigateur, basé sur le langage ChucK. Il illustre les avancées permises par l'exécution dynamique de langages spécialisés dans un contexte Web.

Dans la même lignée, Web Audio Modules (WAM) propose un standard pour le développement de modules audio réutilisables dans des hôtes Web. Inspiré des plugins VST, l'écosystème WAM facilite la création collaborative et la réutilisation de traitements audio complexes dans un cadre ouvert. Cette approche permet de créer des environnements de production musicale complets directement dans le navigateur, comme le démontrent des projets comme Sequencer Party ou les hôtes compatibles WAM.

Enfin, une étude comparative récente entre applications mobiles natives et Progressive Web Applications met en évidence la maturité croissante des technologies Web pour les usages nécessitant des performances proches du temps réel. Les avantages en termes de portabilité, d'accessibilité et de maintenance allègent considérablement les contraintes imposées par les écosystèmes natifs.

3. Des applications pour la transmission

3.1. Le projet SmartFaust

Imaginé et développé par Christophe Lebreton en 2013, SmartFaust ^{4 5} est un concept d'applications musicales pour smartphones développées avec FAUST pour iOS et Android. Ces applications autonomes ont la particularité de faire uniquement appel aux gestes de l'utilisateur et non pas à un pianotage sur l'écran de l'appareil. Elles exploitent les capteurs de mouvement de ces appareils afin d'en faire de véritables instruments de musique.

En mars 2014, lors de la Biennale Musique en Scène, GRAME organisait un concert d'un nouveau genre qui allait faire date. S'appuyant sur les applications SmartFaust, il est imaginé un grand concert participatif aux Subsistances à Lyon sous la conduite du compositeur Xavier Garcia.

Le concert s'est déroulé en deux temps, avec en premier des pièces pour instrumentistes solistes et un « chœur » de smartphones constitué de volontaires, musiciens ou non, préalablement sensibilisés à l'occasion d'un stage avec le compositeur, et en deuxième une

⁴ <https://www.grame.fr/articles/smartfaust>

⁵ <https://github.com/grame-cncm/smartfaust>



pièce participative pour le public. La particularité de cette deuxième partie reposait sur le caractère instantané de la création, sans répétitions préalables ni prérequis musical. Seuls un smartphone et un « leader » étaient nécessaires pour la réalisation collective de cette expérience musicale.

Ce concert marqua le début et l'intérêt de ces applications comme outil au service de la transmission des musiques électroacoustiques et de l'informatique musicale. En effet, la palette de sonorités proposées par ces applications est directement issue de l'esthétique électroacoustique. Si ces sonorités peuvent surprendre le public non averti par leur caractère inouï et quasi science-fictionnel, le caractère ludique prend le dessus et amène les participants à se familiariser avec.

3.2. Développement et autres projets

Depuis 2014, de nombreux autres projets utilisant cette même technologie ont été développés. On y retrouve la série d'applications SmartFaust, mais également les Geek Bagatelles et les GameLan, des variantes plus esthétiques que technologiques puisque le concept et l'utilisation restent similaires. L'accessibilité et l'adaptabilité de ces applications en font un instrument idéal pour un large panel de participants. Du scolaire aux adultes amateurs en passant par les élèves et étudiants des établissements d'enseignement artistique spécialisés et les musiciens professionnels, une multitude d'expériences solistes et collectives ont pu être réalisées.

Voici quelques exemples de projets menés pour les smartphones :

- **Art by Smartphones**, de Xavier Garcia avec le public des TER de la région Auvergne-Rhône-Alpes
- **INDUS 4.0** et **Taylorisme**, deux pièces pour orchestre de smartphones du diptyque «Retour vers le futurisme» créées avec et par les élèves en CM2 de l'école Aveyron (Lyon 1er)
- **À vos smartphones, prêts...jouez !**, projet du percussionniste Roland Merlinc avec les élèves en CAP du lycée professionnel Saint-Marc (Lyon 2e)
- **Virtual Rhizome**, pour soliste et 2 smartphones, du compositeur Vincent Carinola⁶
- **Smartland Divertimento**, installation de téléphones, par Stéphane Borrel et Christophe Lebreton⁷
- **Gamelan**, 2^o jeu d'applications basées sur des notions de son instrumental et utilisant pour plusieurs d'entre elles des sons enregistrés, par Romain Constant et Elodie Rabibisoa⁸
- **Stages de médiation par la musique** avec les travailleurs sociaux de l'ARFRIPS
- **Travail de recherche et création** du groupe 4EDED de la Maîtrise de la Loire

4. Développement des applications

Les applications SmartFaust ont été développées au départ sous la forme d'applications natives sur iOS et Android. Les contraintes étaient d'avoir de très bonnes performances audio

⁶ <https://www.grame.fr/productions/vincent-carinola-virtual-rhizome>

⁷ <https://www.grame.fr/productions/stephane-borrel-smartland-divertimento>

⁸ <https://www.grame.fr/articles/gamelan-5d839c31f332c>



temps-réel, avec des latences faibles en particulier dans le traitement des données des capteurs, de façon à offrir une expérience utilisateur optimale en termes de « jouabilité⁹ ».

4.1. Des applications natives

Le développement logiciel « bas niveau » était réalisé de la manière suivante :

- Côté iOS : avec un *backend* audio construit sur CoreAudio¹⁰, le code DSP FAUST exporté en C++, et la partie applicative écrite en ObjectiveC et les interfaces de programmations d'Apple
- Côté Android : avec un *backend* audio construit avec Oboe¹¹, le code DSP FAUST exporté en C++, et la partie applicative écrite en Java (et interaction avec la partie *backend* grâce à JNI (Java Native Interface) et les couches graphiques d'Android
- La récupération des valeurs d'accéléromètre et gyroscope doit être faite en utilisant les interfaces de programmation spécifiques de chaque système d'exploitation.

La publication devait ensuite passer par des plateformes propriétaires : l'App Store pour iOS et le Google Play Store pour Android, processus qui implique une validation souvent longue et parfois contraignante¹². Cette validation a pour objectif de garantir que les applications respectent un ensemble strict de critères, notamment en matière de sécurité et de gestion des données sensibles.

4.2. Maintenance et pérennité

Ces dernières années, plusieurs utilisateurs nous ont fait remonter la disparition de nos applications des plateformes de téléchargement, ce qui a provoqué un arrêt ou une modification partielle des projets artistiques et pédagogiques. En effet, l'évolution régulière des critères d'hébergement sur les plateformes dédiées d'iOS et Android, contraint les développeurs à assurer un suivi afin de garantir la permanence de leurs applications sur lesdites plateformes. Ainsi, au fur et à mesure de l'évolution de la plateforme Web, de l'apparition d'interfaces de programmation plus modernes et plus fiables (comme la Web Audio API¹³), il a été envisagé d'opérer une refonte complète des technologies employées. La migration s'est faite en plusieurs étapes détaillées dans les sections suivantes.

4.3. Vers les applications Web

L'utilisation de plateforme Web présente plusieurs avantages. D'une part, elle permet de mutualiser les efforts de développement autour d'un socle technologique unique, évitant

⁹ La latence de la partie système reste généralement meilleure sur iOS comparé à Android.

¹⁰ https://fr.wikipedia.org/wiki/Core_Audio

¹¹ <https://github.com/google/oboe>

¹² Les applications, conçues pour être utilisées principalement via des gestes et ne présentant pas toujours d'interfaces utilisateur complexes, étaient parfois mal comprises par le service de validation d'Apple, entraînant des échanges fastidieux.

¹³ <https://www.w3.org/TR/webaudio-1.1>



ainsi la duplication du code. D'autre part, le Web offre une portabilité intrinsèque : une même application est accessible sur l'ensemble des systèmes d'exploitation et des appareils, sans dépendre des contraintes spécifiques aux environnements propriétaires. De plus, les standards ouverts et l'évolution rapide des outils Web (WebAssembly, la Web Audio API, PWA) garantissent des performances et des capacités graphiques de plus en plus proches des applications natives, tout en simplifiant la distribution, la mise à jour et l'accessibilité des applications.

En effet, la plateforme Web assure « par construction » un ensemble de garanties en matière de sécurité, qui étaient auparavant vérifiées lors du processus de validation des outils de distribution comme l'App Store ou Google Play Store. Les navigateurs modernes intègrent nativement des mécanismes stricts de confinement (*sandboxing*), empêchant les applications Web d'accéder directement aux ressources sensibles de l'utilisateur, ou au système d'exploitation sous-jacent sans autorisation explicite.

De plus, l'utilisation obligatoire de connexions sécurisées (HTTPS), les politiques de gestion fine des permissions (par exemple, pour l'accès au micro, à la caméra ou au stockage local) contribuent à un environnement sécurisé par défaut. Ainsi, les contraintes de sécurité qui nécessitaient autrefois des vérifications manuelles ou spécifiques aux plateformes mobiles sont désormais garanties directement par l'écosystème Web et son respect des standards.

Enfin, les technologies Web présentent une compatibilité ascendante, ce qui leur permet de fonctionner avec les évolutions futures des standards.

4.4. Le package faustwasm

Depuis 2014, les programmes DSP FAUST peuvent être compilés en code binaire pour être exécutés dynamiquement dans les navigateurs, en utilisant l'infrastructure des nœuds Web Audio pour le rendu audio.

Une première version de ce projet s'était appuyée sur `asm.js`, un langage prototype développé par Mozilla pour optimiser l'exécution de code JavaScript dans les navigateurs. En permettant la compilation de langages comme C ou C++ vers un sous-ensemble strict et typé de JavaScript, `asm.js` (en utilisant le compilateur Emscripten¹⁴) offrait des performances proches du natif pour des applications Web intensives.

À partir de 2017, l'arrivée du langage normalisé WebAssembly¹⁵, a permis de porter l'ensemble de la chaîne de compilation FAUST sur le Web de manière plus stable et plus pérenne.

En 2022, une nouvelle version du compilateur pour le Web, avec les fichiers d'architecture pour Web Audio réécrits en TypeScript, a été publiée sous la forme d'un projet séparé nommé `faustwasm`¹⁶. Elle fournit des enrobages TypeScript/JavaScript pour les codes DSP FAUST, permettant de générer des pages HTML autonomes, des modules JavaScript (incluant le code FAUST sous forme de module WebAssembly et diverses ressources), ou même

¹⁴ <https://emscripten.org>

¹⁵ <https://webassembly.org>

¹⁶ <https://github.com/grame-cncm/faustwasm>



d'intégrer le compilateur libfaust dans des applications nécessitant la compilation et un déploiement dynamique de programmes FAUST.

La bibliothèque `faustwasm` permet ainsi de créer des nœuds audio de synthèse et d'effets, mais également des instruments polyphoniques. Elle peut être utilisée dans des projets basés sur Node.js ou directement dans des navigateurs Web, et est publiée sur NPM¹⁷. Le support MIDI est activé automatiquement lorsque des métadonnées MIDI sont utilisées dans le code DSP pour les nœuds de synthèse et d'effets, et est toujours actif en mode polyphonique.

Les capteurs (accéléromètre et gyroscope) sont pris en charge en utilisant le modèle de mémoire partagée¹⁸, qui optimise les transferts de valeurs des capteurs vers le code qui exécute les calculs audio et minimise les latences. Enfin le modèle de déploiement des applications sous la forme de Progressive Web Applications (PWA) peut être utilisé¹⁹.

4.5. Le modèle «Progressive Web Application»

Introduites en 2015 par Google, les Progressive Web Applications (PWA) sont des applications Web offrant une expérience utilisateur comparable à celle des applications natives.

Leur fonctionnement repose sur un système de gestion de cache et de téléchargement des ressources via un Service Worker. Lors du premier accès à l'application, le service est activé et télécharge l'ensemble des ressources nécessaires en les stockant en local.

Une fois cette étape complétée, l'application peut fonctionner même sans connexion réseau. Lors des lancements suivants, elle charge les ressources directement depuis le cache, garantissant ainsi rapidité et fluidité d'utilisation. La structure minimale est la suivante :

- Un manifeste Web (`manifest.json`), fichier JSON qui définit le nom, les icônes, les couleurs et le mode d'affichage de l'application
- Le code JavaScript Service Worker, script qui intercepte les requêtes réseau et met en cache les ressources pour une utilisation hors ligne
- L'application est délivrée par un serveur HTTPS avec un fichier HTML principal qui active le Service Worker

La mise à jour des applications peut être automatique si le mécanisme de Service Worker vérifie la présence de nouvelles versions lors d'une nouvelle connexion réseau et met à jour la version locale²⁰.

4.6. Les Progressive Web Applications FAUST

En combinant FAUST avec le modèle Web Audio et d'autres technologies pure Web, des algorithmes audio optimisés peuvent être exécutés directement dans le navigateur. Cela assure la portabilité et garantit des performances en temps réel essentielles pour les applications musicales et interactives. En accédant aux capteurs, des interactions immersives où les

¹⁷ <https://www.npmjs.com/package/@grame/faustwasm>

¹⁸ Si elle est disponible, distribué par un serveur HTTPS et sur des OS récents.

¹⁹ https://en.wikipedia.org/wiki/Progressive_web_app

²⁰ Une alternative à ce mécanisme parfois compliqué est de simplement désinstaller l'application locale avant d'installer la version plus récente.



mouvements de l'utilisateur servent à modifier dynamiquement les paramètres audio vont pouvoir être décrites.

4.6.1. Génération des applications

Un fichier d'architecture écrit en JavaScript utilise le package `faustwasm` pour charger, compiler le code `wasm`, le déployer sous la forme d'un nœud compatible Web audio, ainsi que le composant `faust-ui`²¹ qui prends en charge l'interface graphique générée automatiquement (le même composant est également utilisé dans l'application Faust IDE²²), tout en prenant en charge l'infrastructure nécessaire au déploiement de l'application en mode PWA.

À partir d'un fichier DSP, le script `faust2wasm` effectue la compilation du code vers le fichier binaire `wasm`, la description du programme sous la forme d'un fichier JSON, ainsi que le fichier d'architecture JavaScript associé.

Après avoir installé le package `faustwasm` en local, un programme `foo.dsp` est alors compilé avec la commande suivante :

```
node faust2wasm.js foo.dsp foo -pwa
```

Ce service de compilation est également disponible dans les outils de prototypage Web comme le Faust IDE. Il est ainsi possible d'exporter directement en choisissant la plateforme « web » et l'architecture « pwa », permettant une phase de test immédiate, notamment sur un smartphone, via un QR code généré pour une installation simplifiée.

Pour le déploiement final, les ressources nécessaires (fichiers `wasm`, manifeste, interface graphique, etc.) peuvent être produites en sélectionnant l'architecture « `webaudiowasm` » et installées sur un serveur Web configuré en HTTPS, condition indispensable pour garantir la sécurité et l'accès aux capteurs.

Les contrôleurs générés par FAUST peuvent être associés aux accéléromètres et gyroscopes grâce à l'usage de métadonnées spécifiques²³ qu'il suffit d'insérer dans la description des labels, comme illustré ci-dessous :

```
...  
accelx    =    vslider("accx    [acc :    0    0    -10    0    10]"  
    ,0,-100,100,1) ;  
accely    =    vslider("accy    [acc :    1    0    -10    0    10]"  
    ,0,-100,100,1);  
...
```

Les paramètres des accéléromètres et gyroscopes sont alors décodés au chargement de l'application et utilisés pour mettre en place des fonctions de mappage entre les valeurs reçues en temps-réel lors des mouvements, et l'intervalle de variation des paramètres musicaux contrôlés²⁴. Enfin l'utilisation de quelques lignes de code spécifiques pour demander explicitement l'autorisation d'utilisation des capteurs est nécessaire sur iOS²⁵.

²¹ <https://github.com/Fr0stbyteR/faust-ui>

²² <https://faustide.grame.fr>

²³ <https://faustdoc.grame.fr/manual/syntax/#sensors-control-metadatas>

²⁴ Un geste de rotation complet sur l'axe des X peut par exemple contrôler la fréquence d'un oscillateur entre 200 Hz et 1000 Hz.

²⁵ C'est la seule spécificité liée au système d'exploitation que nous avons eu à gérer.



4.6.2. Publication des applications

Une fois le code source écrit et compilé, les fichiers générés (le fichier `wasm`, les assets graphiques, les fichiers JSON décrivant les paramètres, ainsi que le manifeste PWA) sont organisés dans un répertoire dédié, ensuite hébergé sur un serveur Web configuré en mode HTTPS. Ce mode est requis par la plupart des navigateurs modernes pour permettre l'installation et l'exécution des PWA, notamment pour garantir la sécurité lors de l'accès aux capteurs du terminal (accéléromètres, gyroscopes, micro, caméra, etc.).

Un fichier `index.html` préconfiguré et les fichiers JavaScript associés assurent le chargement correct du moteur audio et de l'interface utilisateur. Un QR code pointant vers l'URL de l'application sera généré, facilitant ainsi l'installation de l'application sur un smartphone ou une tablette simplement en le scannant.

5. Portage avec le modèle PWA

Les applications historiques ont donc été portées vers la nouvelle architecture PWA en 2024, ce qui a permis de tester, déboguer et valider ce nouveau modèle.

5.1. Développement

Le code FAUST existant a pu être directement réutilisé, avec seulement une adaptation nécessaire pour l'utilisation de fichiers audio, initialement chargés en 2013 comme des ressources externes en C, désormais basée sur la primitive `soundfile` intégrée dans la plateforme Web.

5.2. Publication

La publication des nouvelles applications fin 2024 a été réalisée grâce à l'infrastructure offerte par le modèle GitHub Pages²⁶ avec un site développé en open-source avec l'outil Mkdocs²⁷. Lorsqu'ils sont utilisés dans les applications, les fichiers audio doivent être manuellement ajoutés aux ressources générées par la phase de compilation du code DSP.

Le site FAUST `faustpwa.game.fr` a alors été déployé pour héberger les deux projets historiques de GRAME, SmartFaust et GameLan. Les QR code correspondants à toutes les applications sont automatiquement générés. Actuellement, aucun système permettant de suivre les statistiques de téléchargement n'a été mis en place²⁸. Voici une version PWA (Figure 1.) d'une des applications du projet SmartFaust.

²⁶ <https://pages.github.com>

²⁷ <https://www.mkdocs.org>

²⁸ Un formulaire à remplir explicitement pourrait être utilisé.

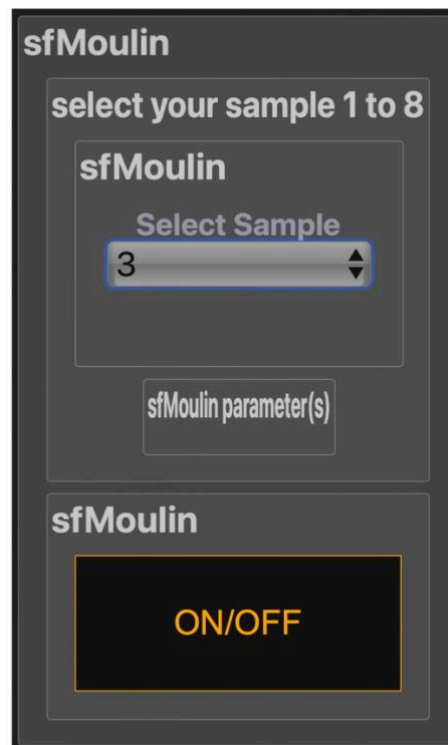


Figure 1 : « sfMoulin » du projet SmartFaust au format PWA, sélection d'un son dans une liste et jeu avec transposition contrôlée par le mouvement.

6. Perspectives

Le développement des PWA offre de nouvelles perspectives d'utilisation et relance l'intérêt pour les applications musicales auprès de la sphère artistique et pédagogique.

Depuis l'automne 2024, les FAUST PWA sont présentés par l'équipe de GRAME aux professionnels, musiciens intervenants, artistes interprètes, compositeurs, enseignants, travailleurs sociaux, animateurs culturels et au sens large, professionnels de la médiation, de l'action culturelle, de la transmission. Ces temps de découverte des PWA permettent à la fois de faire remonter des expériences utilisateur tout en suscitant des idées de projets intégrant les FAUST PWA.

6.1. Nouveaux projets

Citons à titre d'exemples deux projets :

- Ateliers joue avec ton téléphone, programmés en avril 2025 dans le cadre du festival «la semaine Buissonnante» par le Conservatoire de Bourg-en-Bresse. Adressés aux élèves et à leurs parents, ils visent à sensibiliser aux nouvelles technologies et à leurs applications dans un contexte musical. Avec ces ateliers organisés dans le hall et empruntant aux formes de happening, les téléphones sont une porte d'entrée intéressante pour attirer les oreilles et désacraliser les musiques électroacoustiques.
- Commande pour 15 téléphones et 30 voix d'enfants, est une commande de GRAME à la compositrice Emmanuelle Da Costa et au musicien intervenant Romain Constant,



mobilisant les Faust PWA (série GameLan) dans un contexte de chant choral. L'expérimentation des liens entre voix et gestes du smartphoniste se fait en partenariat avec la Maîtrise de la Loire et a pour but de nourrir le travail de composition et de développement des applications GameLan. L'enjeu est de travailler la complémentarité entre les matériaux sonores et la faisabilité technique de la coordination vocale et gestuelle.

6.2. Avantages et contraintes observées dans l'utilisation pratique des applications

Concernant les PWA, la gratuité permet à nombre de professionnels des secteurs de l'éducation, du social, du médical ou des arts de s'en emparer sans contrainte budgétaire. Dans leur utilisation pratique, l'accès à l'application via un simple lien ou QR code, facilite la découverte, l'écoute et l'expérimentation de chaque application en amont d'une séance ou lors d'un atelier. L'utilisation est ainsi simplifiée au regard des applications natives.

On observe d'ailleurs une plus grande utilisation des smartphones personnels lors des ateliers, cela peut s'expliquer à la fois par les évolutions des pratiques numériques, mais aussi par le caractère moins « intrusif » des PWA. À la différence des applications natives, les participants sont libres de créer un raccourci sur leur écran s'ils souhaitent sauvegarder l'accès à certaines pages Web.

Quant à l'interface des FAUST PWA, simplifiée et réduite à quelques commandes (*ON/OFF, samples...*), elle permet la compréhension et la prise en main rapide pour tous. Les facilités d'utilisation des FAUST PWA sont également une amélioration pour les artistes qui encadrent les projets et animent les ateliers. L'expérimentation étant plus aisée et plus rapide, les intervenants peuvent consacrer plus de temps à la sélection d'applications adaptées à l'âge ou aux spécificités de chaque public.

Par ailleurs, le temps de préparation de chaque smartphone en amont ou début d'atelier est considérablement réduit. Chaque participant peut gérer en quasi-autonomie l'ouverture de l'application souhaitée en flashant le QR code correspondant ou en ouvrant le lien FAUST PWA.

6.3. Apprentissages par le jeu

Avec ces applications, le smartphone devient un instrument à part entière. Écran, claviers, réseau mobile, internet, communications diverses sont mis de côté pour laisser place à la musique et au jeu. L'absence d'interface graphique permet la libération de l'oreille et de la main, tout comme un musicien ne regarderait plus ses doigts pour jouer de son instrument.

A l'image des Icebreakers qui permettent à un groupe d'individus de se rencontrer par le jeu et le rire, les applications musicales créent dès les premières minutes de leur utilisation un cadre ludique sans nécessiter de prérequis musicaux. Même si le mouvement demande à s'affiner au fur et à mesure, la concrétisation immédiate du son relatif à une action individuelle ou collective provoque chez n'importe quel participant un plaisir tout similaire au jeu.

Les FAUST PWA sont donc des applications adaptées aux projets d'éducation artistiques et culturels, tout en permettant de sensibiliser aux musiques électroacoustiques et aux technologies de notre quotidien :



- L'écoute de soi et des autres et donc le savoir-être
- La mémorisation de gestes, de sons et de leur enchaînement
- La coordination et la maîtrise de son corps
- Le plaisir de la progression, par la précision apportée aux mouvements
- L'autonomie et la responsabilisation de chacun au sein du groupe, en tant qu'interprète, musicien ou « chef d'orchestre »
- Des notions musicales comme les dynamiques, les *tempi*, les rythmes, etc.

6.4. Créativité

Les Faust PWA et les smartphones-instruments apportent une grande liberté dans le travail créatif. Les participants sont amenés à travailler autant le jeu improvisé que l'interprétation d'un répertoire conçu pour, avec la question des écritures et des notations que cela engendre. L'autonomie de l'instrument et de son musicien donne aussi une grande liberté dans la mise en espace de la création sonore. Les déplacements et la spatialisation induite des sons font partie intégrante des pratiques de smartphonistes. Cette liberté d'espace permet le croisement et la rencontre entre le monde musical et d'autres disciplines scéniques (théâtre, danse, chant...) ou visuelles (vidéos, performances...).

Au-delà des limites du smartphone, la grande diversité d'objets sonores proposée avec les Faust PWA ouvre le champ à une large exploration d'enchaînements et de polyphonies possibles. Ces temps d'exploration et d'invention peuvent d'ailleurs être tout ou partie d'un projet participatif.

La pratique des Faust PWA peut également donner lieu à des travaux parallèles comme l'initiation à la programmation en Faust, l'invention d'une notation spécifique à l'écriture d'une pièce pour smartphones ou bien même un travail plastique et de design pour customiser l'objet-instrument.

6.5. Contraintes et limites des FAUST PWA

En tant qu'utilisateur, le blocage le plus conséquent que l'on rencontre est celui du volume sonore limité du téléphone. Des tests ont été menés pour amplifier le volume en connectant le smartphone à une enceinte Bluetooth. Les enceintes ne pouvant pas être programmées pour un téléphone donné, le Bluetooth n'est possible que pour l'utilisation simultanée d'un ou deux appareils maximum dans le même espace.

Toujours dans la recherche d'une amplification des smartphones, des mini enceintes ont été connectées en filaire. La gestion de l'amplification par chaque smartphoniste est alors plus facile à mettre en place. Reste l'option d'amplifier l'ensemble des smartphones, la solution la plus qualitative d'entre toutes, mais qui nécessite des moyens supplémentaires matériels (microphones et diffusion sonore), humain (sondier), temporel (temps de répétition dédié) et donc financier, moyens peu réalistes au regard du cadre des actions culturelles. Une solution reste encore à tester, celle de l'impression 3D de pavillons à accrocher sur les smartphones.

D'autre part, les smartphones n'échappent pas à la question de la durabilité des technologies. Ceux devenus obsolètes ne peuvent plus être mobilisés pour les Web App. Dans certains contextes la question du matériel pose donc problème. Par exemple, en écoles



primaires et collèges, les élèves n’ont pas toujours de téléphones personnels, et ceux récupérés auprès des parents ont bien souvent des versions trop anciennes pour exécuter les FAUST PWA.

D’une manière générale le cadre contraint de l’éducation nationale peut être bloquant pour l’utilisation des WebApp : pas de wifi dans les établissements (nécessaire pour l’ouverture des pages Web), smartphones interdits dans certains établissements et quand ils sont autorisés, le caractère personnel de l’objet peut créer des disparités et des inégalités entre les élèves.

Peu de solutions existent, mis à part le prêt d’un stock de smartphones par GRAME pour les projets à proximité et la sensibilisation des professionnels de l’éducation aux PWA.

Malgré ces contraintes, l’expérience acquise dans l’usage des technologies Web laisse penser que la pérennité des applications sera plus facile à garantir sur le long terme.

7. Remerciements

Plusieurs développements présentés dans ce travail, notamment l’implémentation du mode PWA, la gestion fine des capteurs de mouvement (accéléromètre et gyroscope), ainsi que l’intégration avec l’infrastructure Web Audio, ont été réalisés en étroite collaboration avec Shihong Ren.

En tant que principal architecte du projet FAUST IDE, Shihong a joué un rôle important dans la modernisation de l’écosystème Web de FAUST et son utilisation plus simple dans les projets de transmission présentés.

Ce projet a été partiellement financé par Metamorphoses, projet européen Erasmus+ KA2 No 2022-1-FR01-KA220-VET-000085833 pour la formation professionnelle²⁹.

8. Conclusion

Le passage des applications FAUST vers le Web, via l’utilisation des technologies WebAssembly et du modèle PWA, marque une avancée importante pour les utilisations pour la transmission. Cette évolution rend les outils plus accessibles, pérennes et adaptés aux contextes artistiques et pédagogiques, tout en favorisant la créativité, l’inclusion et l’autonomie des participants.

Bibliographie

Berggren, William. “An analysis and comparison of the Native mobile application versus the Progressive web application”, Mid Sweden University, 2023.

Buffa, Michel, Ren, Shihong, Burns, Tom, Vidal-Mazuy, Antoine, Letz, Stéphane. “Evolution of the Web Audio Modules Ecosystem”, WAC 2024 - Web Audio Conference, 2024.

²⁹ Digital aesthetic for the next music training: <https://metamorphosesproject.eu>



Letz, Stéphane, Orlarey, Yann, Fober, Dominique. “FAUST Domain Specific Audio DSP Language Compiled to WebAssembly”, Companion Proceedings of The Web Conference, 2018.

Lindetorp, Hans, Falkenberg, Kjetil. “WebAudioXML: Proposing a new standard for structuring web audio”, Proceedings of the Sound and Music Conference, 2020.

Mulshine, Michael, Wang, Ge, Chafe, Chris, Atherton, Jack, Feng, Terry, Betancur, Celeste. “WebChucK: Computer Music Programming on the Web”, Proceedings of the International Conference on New Interfaces for Musical Expression, 2023.

Robaszkiewicz, Sébastien, Schnell, Norbert. “SoundWorks – A Playground for Artists and Developers to Create Collaborative Mobile Web Performances”, Proceedings of the Web Audio Conference (WAC), Paris, France, 2015.

Roberts, Charles, Wakefield, Graham, Wright, Matthew. “The Web Browser As Synthesizer And Interface”, Proceedings of the International Conference on New Interfaces for Musical Expression, 2013.