Numéros / n° 6 - Techniques et méthodes innovantes pour l'enseignement de la musique et du traitement de signal

« Le Web, une solution ubiquitaire et accessible pour l'enseignement et la pratique temps-réel de la musique »

Thomas Cipierre

Résumé

L'idée principale de ce texte est que le Web pourrait changer les expériences d'apprentissage et les concerts de musique électronique grâce à des applications et des contrôles ubiquitaires. Après une argumentation sur l'intérêt pédagogique d'une approche modulaire, quelques outils actuels du développement Web sont présentés, afin de proposer une base de construction d'applications complexes, les plus pérennes possible.

Introduction

Les outils technologiques de l'informatique musicale ont évolué si rapidement au cours des dernières décennies qu'il nous paraît aujourd'hui normal de pouvoir écouter de la musique en permanence, ou même de produire sa musique. Ce qui n'était qu'un rêve pour beaucoup de compositeurs du xx? siècle ? cette ubiquité musicale et le fait de pouvoir contrôler exhaustivement les paramètres du son ? se trouve maintenant à portée de main. Non seulement nous pouvons jouer en temps-réel les paramètres musicaux suivant des calculs mathématiques avancés, mais bien loin des premiers synthétiseurs de laboratoire onéreux, des premiers ordinateurs et tentatives fastidieuses de détournement de langages de programmation génériques, nous pouvons maintenant créer la musique sur des ordinateurs fixes, portables, des tablettes et des smartphones relativement abordables, à l'aide de logiciels et de langages de programmation dédiés.

L'informatique musicale devrait alors paraître aisément accessible, notamment avec la disponibilité d'interfaces utilisateurs (GUIs) attrayantes et ludiques, de périphériques de contrôles « plug-and-play », de processeurs puissants, mémoires vives et espaces de stockage peu coûteux permettant le temps-réel, de contrôles instinctifs par capteurs électroniques et autres surfaces tactiles. Malheureusement, les choses ne sont pas si simples, tous ces outils ne fonctionnant que dans un contexte informatique spécifique et demandant aux utilisateurs de plonger dans des langages de programmation et des protocoles de communication informatiques pour toute utilisation avancée. L'idée principale de cet article est que le Web pourrait changer les expériences d'apprentissage et les concerts de musique électronique grâce à des applications et des contrôles ubiquitaires. Pour le programmeur, cela reviendrait à coder une seule fois son application et à la rendre directement disponible sur de très nombreux appareils et systèmes d'exploitation (OSs). L'utilisateur, quant à lui, pourrait jouer son application favorite sur de nombreux appareils, sans procéder à une quelconque installation, tout en utilisant les capteurs intégrés et les périphériques de contrôle à sa disposition. Finalement, pour l'artiste, toute production musicale pourrait facilement être collaborative et restituée en interaction temps-réel avec son public.

Dans la transition des ordinateurs personnels, d'abord vers un Internet multi-plateforme et distribué, avec ses réseaux et applications riches [RIAs], et maintenant vers des appareils mobiles et des environnements informatiques multi-utilisateurs et ubiquitaires, nous devons faire face à l'impact de ce medium dans la vie de beaucoup de personnes. De nouveaux dispositifs mobiles encore plus avancés devenant disponibles, il est d'autant plus important d'avoir des outils sophistiqués tout en supportant intuitivement la créativité, en particulier pour permettre aux utilisateurs de participer à des activités musicales, comme la création musicale, la performance, l'expérimentation, etc. (Keller, Lazzarini, Pimenta, 2014, p. xi-xii)

1. Atteindre de nouveaux horizons sonores

Toute phase d'apprentissage passe par la constatation d'une « incompétence consciente » (O'Connor, Seymour, 2011, p. 8), qui, suivant l'état d'esprit de l'apprenant (Dweck, 2006), peut se traduire par une interaction hasardeuse et empirique avec le nouvel outil étudié. Mais que saisiriez-vous si ce nouvel instrument était un logiciel ou un langage de programmation ?

Les instruments acoustiques se montrent ici bien plus pratiques : quelle que soit la manière dont vous les utilisez ? même hors de l'intention première du luthier ? ils émettent un son et nous pouvons en apprendre quelque chose. Il semblerait donc que les ordinateurs génériques devront toujours être préalablement préparés à la production de son, au détriment de la spontanéité, ce qui reste un obstacle à l'apprentissage des musiques électroniques pour beaucoup de musiciens ? un intérêt certain pour la musique n'étant pas forcément synonyme d'un même intérêt pour l'environnement informatique.

Cependant, nous avons pu récemment constater de nouvelles pratiques de base chez le musicien amateur, grâce à l'omniprésence des stations audionumériques de travail (DAWs), permettant l'édition et la production de musique. Un fossé se referme donc pour l'accès aux musiques électroniques, les étudiants n'étant plus restreints par l'argent pour apprendre sur de nombreux modules physiques onéreux afin de générer des sons ou de les transformer à l'aide d'effet, et pouvant simplement tester gratuitement une quantité astronomique de plugins téléchargeables sur le Web.

Mais un nouveau problème apparaît : de l'omniprésence des DAWs émerge une forte habitude de contrôle des paramètres musicaux sur une temporalité fixe et linéaire, au détriment des interactions avec des contrôleurs physiques, et plus généralement de la musique temps-réel. Par exemple, lors de mes enseignements auprès d'étudiants en licence première année de musicologie, j'avais simplement assigné les paramètres de temps et de feedback d'un délai aux accéléromètres d'un smartphone pour observer, avec stupéfaction, les étudiants se réjouir autour d'un « nouvel effet » ? alors que je pouvais voir certains d'entre eux posséder une version physique du même effet sur un pédalier de guitare électrique, et que d'autres en avaient largement usé sous forme de plugin dans un DAW. La syntaxe peut changer considérablement ce que nous pouvons faire dans un même contexte sémantique [(1)], et j'ai réalisé ce jour-là que les interfaces de contrôle se comportaient de la même façon vis-à-vis du contexte de l'effet. L'enseignant, en tant que designer et dans son choix des outils musicaux proposés, a donc un impact significatif sur les expériences d'apprentissages et l'accès à de nouveaux horizons sonores.

À l'état le plus fondamental, c'est la nature des relations de travail établient entre les compositeurs et les interprètes et leurs outils de production sonore qui détient la clé de l'échec ou de la réussite. Ces relations sont en fin de compte dépendantes des modes de communication et d'interaction qui peuvent être facilités par les technologies disponibles, reliant les mondes de la créativité et de la subjectivité avec l'environnement hautement objectif de l'ingénierie scientifique. (Manning, 2012, p. 8)

Des environnements et des langages de programmation pour la musique temps-réel existent évidemment, mais leur usage présuppose toujours de nombreuses aptitudes techniques pour les faire opérer et communiquer entre eux, même si nous demandons seulement dans un premier temps aux étudiants d'utiliser des instruments et effets préconçus. Pour la plupart des apprenants, nous abandonnons ainsi une approche pratique et efficace dans un environnement familier ? un DAW ? pour parfois plusieurs années d'apprentissage technique autour de la programmation informatique, d'autant plus difficile à saisir que les étudiants n'ont pas mentalement abstrait et connecté ces outils avec l'expressivité musicale.

Un autre problème épineux se cache sous les logiciels propriétaires souvent choisis par les enseignants ou les étudiants eux-mêmes, simplement parce qu'ils sont largement employés dans le monde professionnel.

Accepteriez-vous d'apprendre avec assiduité un instrument acoustique, disons durant une année, composant plusieurs pièces, pour être ensuite forcé de le remplacer ? non seulement pourrait-il avoir une nouvelle forme, accordage ou méthode d'excitation, mais il pourrait également sonner différemment ? Vous devriez oublier vos réflexes, retravailler toutes vos compositions, voire en laisser tomber quelques-unes, car elles ne seraient tout simplement plus jouables sur ce nouvel instrument. Cela paraît invraisemblable ? les instruments acoustiques n'évoluent pas si rapidement, et personne ne peut vous interdire d'en jouer un quel qu'il soit. Seulement, les ordinateurs, eux, peuvent le faire ! Ils évoluent constamment pour faire face aux changements de matériel, des OSs et des logiciels se battant sur les routes sinueuses de l'économie capitaliste actuelle, de la sécurité (ou malheureusement de l'exploitation) des données personnelles et des nouvelles demandes et nouveaux comportements sociaux. Ces évolutions peuvent vous empêcher de jouer vos créations numériques, car les versions de logiciel ne sont pas toutes maintenues avec les changements de version des OSs pour des raisons financières, ce qui, pour la grande majorité des logiciels propriétaires utilisés, empêche une mise à jour par une des nombreuses communautés enthousiastes et dévouées du monde libre. Plusieurs chefs-d'oeuvre de l'histoire de la musique électronique se retrouvent ainsi cantonnés aux archives, étant dans l'impossibilité d'utiliser le matériel et les versions logiciels nécessaires à leur reproduction. Des formalisations ont été amorcées dans l'espoir de documenter le processus créatif et d'interprétation de la musique informatique, sous une forme suffisamment abstraite pour se débarrasser des outils utilisés par le compositeur, mais la musique résiste encore à sa diffraction en petits éléments autonomes à même de la décrire en termes de créativité et de sensation subjective.

[...] Bon nombre de problèmes clés en termes d'art et de pratique de la musique électronique demandent toujours à être résolus, démontrant que des avancés technologiques ne résultent pas nécessairement en des améliorations concomitantes dans leurs applications créatives. Il n'y a, par exemple, toujours aucun langage universel pour exprimer les idées musicales dans un format ayant une équivalence directe avec les ressources techniques qui sont nécessaires pour les réaliser. Cela crée beaucoup de difficultés spécifiques au contexte, qui doivent être adressées, et les progrès des dernières années dans ce domaine sont loin d'être écrasants. (Manning, 2013, p. viii)

Je pense que les enjeux sont ici nombreux et existentiels. Du point de vue musical, il paraît évident que les outils sont les catalyseurs de l'expression du compositeur (Schaeffer, 1966, p. 16), que ce soit pour générer ou transformer le son. Plus ils seront nombreux et variés, plus le compositeur pourra choisir avec finesse ce qui lui conviendra. Du point de vue de l'individu et d'une société, il est de ma conviction personnelle que les Arts devraient avoir une part essentielle dans tout environnement d'apprentissage, quelle que soit la filière, l'école, ou la forme d'apprentissage autonome. Il me paraîtrait important d'abolir les frontières idéalisées de la rigueur scientifique et son objectivité axiomatique, en ajoutant une valeur sensible par l'intégration des Arts à tout cursus d'apprentissage, à pied d'égalité avec les sciences dites dures.

En termes d'enseignement, le mouvement des standards favorise une instruction directe d'informations, de compétences factuelles et d'enseignement par classe plutôt que par groupes d'activités. Il est sceptique envers la créativité, l'expression personnelle, et les modes de travail non-verbaux, non-mathématiques, d'apprentissage par la découverte et de jeux imaginatifs, ce même dès la maternelle. (Robinson, 2015, p. 14)

À travers la musique, la danse, les arts visuels, le théâtre, et le reste, nous donnons formes à nos émotions et nos pensées sur nous-mêmes, et comment nous vivons le monde qui nous entoure. Apprendre ce qu'il y a dans et autour des Arts est essentiel pour le développement intellectuel. [Robinson, 2015, p. 90]

2. Essayer de trouver un compromis

Tous ces problèmes ne peuvent pas être adressés pleinement avec notre dispositif technologique actuel, mais j'espère qu'en partageant mes réflexions actuelles, mes développements et enseignements, je pourrai contribuer à la construction d'un chemin vers l'accessibilité, la spontanéité, les apprentissages autonomes et les développements informatiques pérennes.

2.1. L'accessibilité par abstraction personnelle et apprentissage empirique

Il y a un grand dilemme dans la construction d'outils musicaux à buts éducatifs. Nous devrions rechercher avant tout de réels résultats musicaux, c'est-à-dire sans trop restreindre les paramètres musicaux, la créativité et l'expression étant souvent atteintes dans la précision et les petites variations (Pinch & Trocco, 2004, p. vi), tout en essayant d'abstraire au maximum ces mêmes paramètres (ou groupes de paramètres), suffisamment dissociés les uns des autres, afin d'être compris par les étudiants.

Le design paramétrique est un processus analytique dans lequel quelqu'un envisage les influences individuelles et corporelles de tous les paramètres d'un son existant ou imaginé. Il apprendra que les activités paramétriques ne sont pas isolées les unes des autres. La hauteur influence l'intensité, l'intensité influence le timbre, le timbre, en retour, influence la hauteur, et le cycle continue. Ainsi, en décidant comment obtenir le résultat désiré à partir d'un instrument électronique, la personne doit être familière avec les rôles et limitations de chaque partie de l'instrument, tout en possédant une compréhension de base sur la nature psycho-acoustique de ce qu'elle recherche. (Strange, 1983, p. 4)

L'abstraction est un processus cognitif personnel et complexe (Aleksandrov, 1999, p. 8) nous permettant de comprendre notre monde et de survivre ? c'est pour cela que nous abstrayons notre entourage dès notre plus jeune âge (Holt, 1982, p. 46-48). Seulement, même si les outils et symboles disponibles permettent de véhiculer des abstractions communes à un groupe social, sans avoir à toujours « réapprendre de zéro », il ne faut pas négliger l'inertie des concepts de haut niveau récents, demandant parfois quelques décennies pour entrer dans un subconscient collectif au monde de la recherche.

Il est important de remarquer que le concept de nombre, qui a été travaillé avec tant de difficultés sur une longue période de temps, est aujourd'hui maîtrisé par un enfant avec une aisance toute relative. Pourquoi ? La première raison est, évidemment, que l'enfant entend et voit des adultes manipulant constamment ces nombres, lui apprenant même à en faire tout autant. Mais la deuxième raison, et c'est celle sur laquelle j'aimerais attirer précisément votre attention, est que l'enfant a déjà à portée de main les outils et symboles pour les nombres. Il apprend d'abord ces symboles externes comme nombres et seulement ensuite en maîtrise la signification. (Aleksandrov, 1999, p. 11)

Il convient donc de veiller à ne pas perdre l'attention des étudiants sur des sujets essentiels en sautant trop rapidement vers des abstractions de haut niveau. Cela veut dire que nous devrions considérer chaque étudiant dans son individualité (O'Connor, Seymour, 2011, p. 4) et proposer des outils pouvant répondre à un intérêt général en restant adaptables. C'est pourquoi, j'ai la conviction qu'une approche d'enseignement empirique est essentielle pour obtenir un apprentissage efficace et durable (Robinson, Aronica, 2015, p. 77). Sans entrer dans le concept de classe inversée (Robinson, Aronica, p. 74-75) qui pourrait si bien l'accompagner, elle autorise des plages de temps et de plaisir préalables à l'ancrage de concepts qui ont dès lors un but, un geste, un son ou un nom, pouvant même entraîner le cours vers des idées ou des développements d'outils inattendus, suivant une nouvelle possibilité imaginée ou accidentellement rencontrée par un étudiant (Strange, 1983, p. 5). Il est vrai que l'expérimentation commence par un usage extrême et excessif des paramètres à disposition ? testant de ce fait les limites de nos modèles ? et reste assez rarement musicalement intéressante. Mais elle s'affine très rapidement sur des zones plus restreintes de contrôle, vers de nouveaux comportements musicaux, menés par une envie personnelle de comprendre chaque module et d'avoir une vision plus générale de l'instrument ou de l'effet.

Une de nos manières d'apprendre est par la maîtrise consciente de petites parties de comportement, et leurs combinaisons en groupes de plus en plus larges, afin qu'elles deviennent habituelles et inconscientes. On construit des habitudes afin d'être libre d'observer d'autres choses. [...] Nous vivons dans une culture qui croît que ce que nous faisons le plus est fait consciemment. Pourtant, ce que nous faisons le plus, et que nous faisons de mieux, nous le faisons inconsciemment. [...] L'idée d'être capable de comprendre un monde infiniment complexe avec une pensée consciente ne pouvant seulement contenir sept informations simultanées, est évidemment ridicule. (O'Connor, Seymour, 2011, p. 6-7)

Les expérimentations empiriques seules sont encore loin de donner une base théorique stable et une compréhension vaste de l'objet de l'étude, et l'étudiant ne pourra sûrement pas reconstruire tout le processus qu'il a traversé pour arriver à son résultat. Cependant, il peut maintenant relier de nouvelles théories aux sons qu'il a créés, à une expressivité qu'il a pu explorer selon son propre jugement subjectif. Ainsi, les paramètres musicaux, que nous essayons en premier lieu de faire constater à l'étudiant comme une « incompétence consciente » (O'Connor & Seymour, 2011, p. 8) doivent être modularisés, ce qui est commodément analogue aux formes des premiers synthétiseurs modulaires. C'est pour cela que j'essaie de toujours fournir aux étudiants des versions numériques imitant (et non forcément simulant) certains de leurs modules, afin d'autoriser des matriçages expérimentaux.

Nous avons, durant la dernière décennie, observé un grand intérêt pour la programmation par schémas

fonctionnels (dits « blocs-diagrammes ») à travers les langages de programmation visuels (VPLs) ? comme Pure Data ou Max pour le domaine spécifique de la création musicale. Pour l'informatique musicale, ils sont des outils intéressants pour tester des programmations expérimentales et obtenir assez facilement des résultats musicaux (Puckette, 2007, p. ix). Encore une fois, cette sémantique n'est pas restreinte à la syntaxe des VPLs et pourrait facilement s'étendre à d'autres façons de programmer dans un futur proche (Orlarey, 2009, p. 362). Évidemment, plus les modules deviennent complexes et nombreux, plus les contrôles se doivent d'être pluriels et précis, résultant rapidement dans d'immenses flux complexes de données qui ne peuvent être facilement gérés graphiquement (Pottier, 2009, p. 232). Mais en ce qui me concerne, la quantité de travail à fournir pour rendre accessible ces outils est largement suffisante pour ne pas avoir à se préoccuper des techniques de composition avancées dans un premier temps.

Gérer les concepts d'organisation paramétrique, du « voltage control », et tracer des patchs de connexion est sûrement suffisant pour le débutant, sans avoir à être concerné par les grâces du geste compositionnel, du moins dans les premiers stades. [...] Tous ceux qui sont impliqués dans les musiques électroniques ne sont pas forcément intéressés par la composition. Et je suis assez sûr que plus les interprètes des instruments électroniques seront compétents, plus les compositeurs de musiques électroniques seront heureux. (Strange, 1983, p. 3-4)

2.2. Des environnements de création pérennes

Nous avons vu précédemment qu'aucun outil n'existe actuellement pour aborder complètement le problème de la pérennité des programmes d'informatique musicale ? je ne peux donc garantir que la démarche qui suit permettra aux étudiants de jouer leurs créations, disons dans une décennie. Cependant, je suis persuadé que c'est un exemple de la façon dont nous devrions procéder, ou du moins un bon départ de réflexion dans l'attente de ces fameux et éventuels outils pérennes. L'idée est de ne pas orienter nos programmes vers une plateforme ou un environnement spécifique, mais de formaliser les idées musicales sous une forme mathématique abstraite ? assurant ainsi précision et déterminisme à nos modèles ? tout en autorisant le contrôle des paramètres à travers des GUIs abstraites. Voici le constat de quelques années de pratiques avec le langage de programmation Faust.

Faust (Orlarey, Fober & Letz, 2009) est un langage de programmation développé par GRAME (Lyon), synchrone et purement fonctionnel ? ce qui, suivant la relation syntaxe/sémantique précédemment citée, est déjà en soi un atout non négligeable face à la majorité des langages de programmation impératifs ?, ayant pour but de fournir une notation adaptée à la description de processeurs de signaux (DSPs) par des fonctions mathématiques. Il s'agit justement de ce premier pas concret vers un développement pérenne, puisque nous pouvons commencer par implémenter nos idées de façon déclarative et par la suite choisir un environnement spécifique pour sa réalisation. De plus, nous pouvons même obtenir automatiquement une documentation mathématique exhaustive de notre DSP, sans aucune référence à une primitive du langage Faust (Barkati & Orlarey, 2011). Ainsi, si vous avez développé votre DSP pour des étudiants dans un certain contexte ? disons par exemple le Web ou Pure Data ?, ces derniers pourront également le compiler vers un nouvel environnement plus adapté à leurs habitudes de travail ? comme un plugin pour DAW. GRAME propose même un compilateur en ligne pour Faust très utile et efficace, évitant aux étudiants de s'occuper de gérer toute dépendance ou kit de développement logiciel (SDK) sur leurs machines (Michon & Orlarey, 2012).

3. Des outils de développement pour la musique sur le Web

3.1. La Web Audio API

Le Web a rapidement évolué d'une simple forme de document avec hyperlien vers d'impressionnantes applications sur une seule page (SPA) et autres applications Internet riches (RIAs), résultant désormais en une plateforme viable pour l'informatique musicale (Wyse & Subramanian, 2013). La Web Audio API est une interface de programmation open-source pour application JavaScript développée par W3C et conçue pour supporter la basse latence, la lecture à l'échantillon près et le traitement de noeuds (ou

modules) dans un contexte audio (ce dernier fournissant les points de connexion d'entrée et sortie physiques de l'ordinateur pour l'enregistrement et la lecture de musique) ? le tout accessible dans un simple navigateur. Elle est basée sur un graphe de signal, analogue au graphe VPL précédemment cité, et fournit par défaut des noeuds pour générer, transformer et analyser le son en temps-réel grâce à une implémentation sous forme de code natif, optimisé par les fournisseurs de navigateur Internet.

Le navigateur Internet est devenu une plateforme de plus en plus viable pour la création et la distribution de nombreux types d'applications informatiques pour les médias. Il n'est pas surprenant de voir que l'audio a une part importante de ces développements. [...] En éducation, nous pourrions voir le développement aisé de logiciel d'entraînement en informatique musicale pour tous les niveaux [...]. Pour la recherche, les applications Web peuvent permettre de facilement créer des prototypes et démonstrations. Les compositeurs et artistes des médias peuvent également bénéficier de la large portée d'Internet pour créer des oeuvres d'art portables. (Keller, Lazzarini & Pimenta, 2014, p. 111)

Malgré l'état récent et en constante évolution de la Web Audio API, nous pouvons d'ores et déjà observer nombre de projets audio florissant sur le Web. Cependant, quelques problèmes majeurs doivent encore être abordés (Lazzarini, Yi & Timoney, 2015), comme le fait que les noeuds natifs ne soient pas consistants dans leurs réponses spectrales en fonction du navigateur choisi, ce qui s'avère rédhibitoire dans un contexte professionnel. Mais la Web Audio API permet également de créer ses propres noeuds, à travers les ScriptProcessorNodes ? obsolètes, mais encore largement utilisés ? ou les WebAudioWorkers qui sont censés les remplacer.

3.2. Faust, FaustLib et Asm.js

Faust permet de compiler un programme DSP en Asm.js, un sous-ensemble JavaScript typé à bas niveau et hautement optimisable, en passant par Emscripten ou directement par le backend Asm.js de la libfaust (Letz, Denoux, Orlarey & Fober, 2015). Asm.js offre des bénéfices de performances importants ? bien que dépendants de son implémentation par les fournisseurs de navigateurs ? et peut, d'après les benchmarks préliminaires officiels, mener à des programmes seulement deux fois plus lents qu'une compilation native. Les instances de notre DSP audio peuvent ainsi être créées et utilisées comme des noeuds réguliers de la Web Audio API. Nous pouvons même directement écrire du code Faust, le compiler et l'instancier à l'intérieur d'une page Web, grâce à la version libfaust.js du compilateur.

3.3. Front-end et protocoles de contrôle pour l'audio

Un des grands intérêts de JavaScript (et Asm.js) est de pouvoir se passer d'un serveur pour le fonctionnement de l'application et de directement faire tourner le calcul dans le navigateur de l'utilisateur ? ce qui, dans le cas contraire, résulterait en de fortes latences pour charger et télécharger les flux de données. Ce calcul front-end (ou côté client) est utilisé par la Web Audio API et par les compilations Asm.js de Faust, mais nous devons tout de même nous préoccuper de la création des GUIs, à moins de se contenter de la génération automatique proposée en option par le compilateur Faust.

JavaScript a attiré énormément de développeurs, et on souligne parfois qu'il est relativement accessible et aisé de l'apprendre. Je ne partage cependant pas ce point de vue, car il est assez facile, pour qui ne maîtrise pas parfaitement la syntaxe et les tournures particulières de JavaScript, de se perdre dans son environnement non typé et d'observer des effets de bord indésirables ou des fuites de mémoire ? et ce n'est pas en s'appuyant sur les messages cryptiques de son debugger qu'il sera possible de dénouer la situation.

La meilleure solution reste d'apprendre le langage, mais pour assister à un développement rapide auprès des novices, et pour obtenir rapidement des résultats graphiques attirants, de très nombreuses interfaces de programmation pour application (APIs) et autres framework sont disponibles. Il faudra avant tout être attentif au fait que, selon les implémentations et les supports réels des navigateurs, certains risquent de demander beaucoup de ressources pour fonctionner, ou plusieurs secondes pour se charger, ce qui affectera l'expérience utilisateur et pourra même générer des artefacts audio en ce qui nous concerne. Je pense personnellement que les Web Components s'accordent plutôt bien avec le concept de graphe audio, et qu'un framework MVW [(2)] (Model View Whatever) permet de formaliser et debugger plus facilement tout développement de type SCRUD (Search, Create, Read, Update, Delete). Les environnements de

typage pour JavaScript [3] se montrent également plus faciles à maintenir en traitant bon nombre de bugs à la compilation, permettant ainsi de maintenir le développement et de prédire les résultats plus simplement ? Elm [4], d'Evan Czaplicki, en est un représentant efficace, notamment grâce aux optimisations de DOM virtuel, d'autant plus que sa formalisation purement fonctionnelle devrait s'accorder à merveille avec celle de Faust.

Les contrôles pour l'audio sont également accessibles dans les navigateurs, comme pour l'OSC (via les WebSockets), le MIDI (via la Web Midi Api), ou encore le Bluetooth, permettant aux étudiants de brancher des périphériques ou d'utiliser le réseau local pour contrôler leurs applications. Cependant, les implémentations de ces normes dans les navigateurs ne sont pas constantes, et même si quelques « solutions de secours » existent en JavaScript, nous pouvons facilement obtenir des programmes instables ou peu performants, voire des conflits de code.

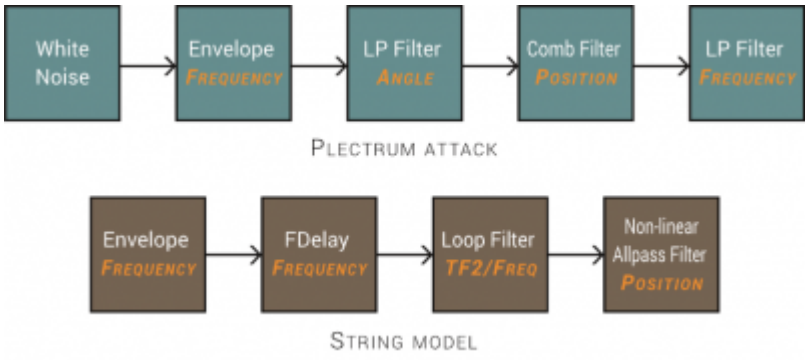
3.4. Le WebHarpsichord ? une preuve de concept

Pendant le projet ANR FEEVER, Laurent Pottier et Luc Faure ont conçu un clavecin temps-réel synthétisé par modèles physiques (Cipierre & Pottier, 2015), à la suite des travaux de portage de la synthèse par guides d'onde vers Faust réalisés par Romain Michon (Michon, 2011), et aux travaux de l'IRCAM sur les modèles de résonances (Potard, Baisnée & Barrière, 1991, p. 135-162). J'ai ensuite été chargé de développer une version polyphonique et temps-réel pour le Web, côté client, afin de proposer une plateforme de production gratuite et facile à utiliser dans un cadre pédagogique ou professionnel [5].

3.4.1. Une rapide vue d'ensemble du clavecin synthétisé par modèles physiques

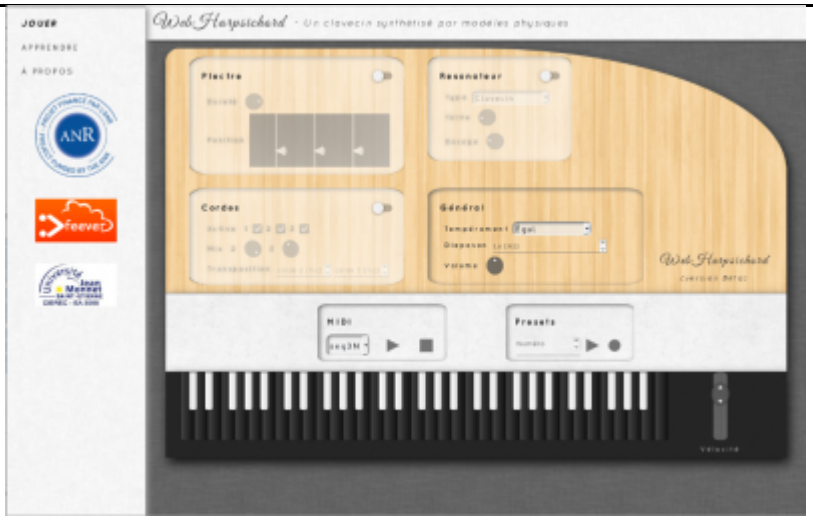
Avoir une idée du traitement de signal codé en Faust permet de réaliser les avancées des technologies Web et les calculs « lourds » qui peuvent fonctionner côté client, grâce aux optimisations de compilation Faust en Asm.js (ou également maintenant en WebAssembly) et à la Web Audio API.

Figure . Modèles de synthèse pour le plectre et la corde



La figure 1 présente les modèles implémentés pour un couple plectre-corde du clavecin. La table d'harmonie étant par contre bien trop complexe pour être modélisée physiquement, elle a été implémentée par modèles de résonance (après avoir enregistré une réponse impulsionnelle sur un clavecin réel). Les trois modèles polyphoniques plectre-corde et les 59 filtres nlf2 [6] nécessaire au résonateur stéréo ont ensuite été intégrés dans une page Web, côté client [7].

Figure . Interface graphique du WebHarpsichord



Comme on peut le voir sur la figure 2, les GUIs permettent de contrôler la dureté et la position d'attaque du plectre, de mixer les niveaux des trois cordes et de modifier indépendamment leurs accordages, de modifier le tempérament général, ainsi que la taille du résonateur et son niveau par rapport au son brut. Il était également possible de désactiver les paramètres avancés pour revenir aux réglages initiaux du clavecin, d'enregistrer et rappeler des sauvegardes de réglages, et de lire des pistes MIDI incluses ? enregistrées par le claveciniste professionnel Martial Morand.

3.4.2. Le Web est-il réellement ubiquitaire ?

Sous mes précédentes références au modèle ubiquitaire des applications Web, pouvant améliorer l'accessibilité des environnements temps-réel pour la musique électronique, se trouve en filigrane un problème majeur. Il est clair que les outils du Web sont en constante évolution et que la complexité de maintenance des applications reviendra aux développeurs, mais tout devrait être transparent pour l'utilisateur et son expérience de l'application, quel que soit le navigateur choisi ? ce qui n'est malheureusement pas le cas. En effet, comme nous avons pu l'entrevoir à travers la réponse spectrale non fiable des noeuds natifs de la Web Audio API, les fournisseurs des navigateurs peuvent implémenter (ou non) les nouvelles spécifications suivant un cahier des charges plus ou moins strict. C'est pourquoi nous avons des consortia tels que W3C, dont le but premier est d'éviter de retomber dans des développements spécifiques à certaines plateformes.

La phase finale de développement du WebHarpsichord a également mis à jour de nouvelles divergences, la complexité du code Asm.js n'étant pas traité de la même façon par Firefox de Mozilla et Chrome de Google, bien que ce ne fût précédemment pas le cas pour du code plus « léger ». De plus, j'ai réalisé les limites des polyfills et autres shims, qui sont basiquement une sorte de « rustine » JavaScript ? temporaire, espérons-le ? pour les navigateurs n'ayant pas encore implémenté les nouvelles fonctionnalités désirées.

Pour résumer les problèmes rencontrés, Chrome délivrait une application avec des artefacts audio inacceptables, surtout au démarrage, mais une interface utilisateur très fluide ? j'avais en effet utilisé Polymer (en version 0.5 à ce moment-là), développé par Google, dans l'espoir de formaliser une première approche Web Components-Faust. Firefox, quant à lui, gérait parfaitement Asm.js ? étant également développé par Mozilla ? et offrait le rendu sonore attendu, mais devait utiliser des polyfills pour la partie GUI, les Web Components n'étant pas encore implémentés. La lourdeur du code JavaScript pour émuler les Web Components apportait alors une latence inacceptable dans la réponse des contrôleurs (et donc dans la génération du son), de l'ordre de la seconde. Concernant les implémentations MIDI, alors seulement accessibles en bêta sur Chrome et via shim sur Firefox, elles étaient tout simplement inutilisables à la suite d'un conflit entre les polyfills des Web Components et les Web MIDI Shim. Une bonne interface mais un son inutilisable sur Chrome, une synthèse parfaite mais une interface inutilisable sur Firefox (sans parler de l'impossibilité d'utiliser le MIDI) sont bien loin de se montrer comme des exemples ubiquitaires du développement Web. On pourrait objecter qu'Asm.js n'est pas un standard W3C, mais c'était à ce moment précis la seule méthode permettant un rendu avancé de DSP en frontend, WebAssembly (ou wasm) n'étant pas totalement formalisé et opérationnel.

4. Travaux connexes

Je vais brièvement présenter ci-dessous quelques applications utilisant la Web Audio API, proches de mes intentions de développement et pouvant se révéler très utile dans un contexte pédagogique.

4.1. FaustPlayground

GRAME a également développé sa propre plateforme Web open-source dédiée à des actions pédagogiques (8). Elle est développée en JavaScript-TypeScript, et permet aux étudiants de jouer en temps-réel de nombreux modules professionnels écrits en Faust, de les éditer directement ou d'en compiler de nouveaux, et de contrôler les paramètres en utilisant les surfaces tactiles ou les accéléromètres de leurs smartphones et tablettes. Finalement, quand les étudiants sont contents de leurs développements, ils peuvent télécharger une application native pour Android ou IOS en flashant un simple QR Code, grâce à l'efficacité de la conversion du matriçage Web Audio des modules Faust en une expression Faust équivalente (Denoux, Orlarey, Letz & Fober, 2016).

4.2. Faust Instant

Michael Bylstra travaille actuellement sur une implémentation efficace du compilateur libfaust.js dans une application Elm (9). Une fois terminée, elle devrait normalement être proposée en open-source et fournir un bel exemple de formalisation purement fonctionnelle Elm-Faust.

4.3. BlokDust

Luke Twyman, Luke Phillips et Edward Silverton ont développé une application Web open-source très attrayante pour la synthèse temps-réel (10). Elle s'intègre parfaitement dans la logique empirique, voire dans ce que certains nomment le « serious gaming » ? les influences du projet viennent en partie de l'aspect « Redstone » du jeu vidéo Minecraft, alors que l'aspect visuel tire ses influences des premiers Lego et de la Reactable. Le projet est codé en JavaScript-TypeScript, affiché par Canvas, et le son est généré par le framework Tone.js (de Yotam Mann) ou des noeuds natifs de la Web Audio API. Le spectre ne devrait donc pas être identique suivant les navigateurs, mais les développeurs encouragent grandement les utilisateurs à utiliser Chrome.

4.4. WebModular

G200kg a développé un synthétiseur modulaire standard et très efficace (11), que j'ai souvent utilisé avec mes étudiants en guise d'introduction à la synthèse. Il est ludique, facile à utiliser, et la génération de musique par notation en Music Markup Language, ainsi que la sauvegarde ingénieuse des patchs dans l'URL, sont un plus non négligeable. Ses développements WebMidiLink apportent également des fonctionnalités expérimentales intéressantes, permettant de contrôler plusieurs synthétiseurs basés sur le Web avec un seul contrôleur.

Conclusion et travaux supplémentaires

Il est indéniable que le Web constitue un environnement d'apprentissage accessible et efficace, voire un outil stable de production et de restitution *live* de la musique. Un travail immense a été fourni pour réduire la distance entre les applications Web et natives, en termes de ressources informatiques et d'expérience utilisateur. Les Service Workers du W3C garantissent, même maintenant, une utilisation offline transparente de l'application ? un chargement préalable de la page dans le cache étant évidemment requis. Malheureusement, les fournisseurs de navigateur n'implémentent pas encore de manière égale et régulière ce qui nous est présenté comme standard, et nous avons toujours un arrière-goût de développement spécifique. Cependant, même si nous devons actuellement recommander, voire au pire des cas obliger l'utilisation d'un navigateur spécifique, nous n'avons jamais été aussi proches d'un

développement ubiquitaire, ce même navigateur étant normalement directement proposé pour les OSs principaux. Finalement, je suis impatient de voir comment les spécifications WebAssembly pourront améliorer ce qu'avait commencé Asm.js, et comment les premières implémentations d'une version multirate de Faust (Orlarey & Jouvelot, 2016) pourraient rendre accessibles les traitements spectraux dans le Web.

---

1. Yann Orlarey fait par exemple référence à la différence entre les symboles de nombre romains et indo-arabes ? même si ceux-là partagent de toute évidence la même sémantique, seul le dernier permet un calcul en donnant systématiquement une signification différente aux symboles en fonction de leur position dans la chaîne numérique. (Orlarey, 2009, p. 335)

2. Par exemple AngularJS, développé par Google : https://angularjs.org/

3. Par exemple Typescript, développé par Microsoft : https://www.typescriptlang.org/

4. http://elm-lang.org/blog/blazing-fast-html-round-two

5. http://musinf.univ-st-etienne.fr/recherches/ClavecinHtml/web-harpsichord.html Malgré une implémentation réussie du moteur Web audio, les développements ont été abandonnés pour le moment, l'interface et les contrôles externes nécessitant d'être recodés de zéro à la suite des conflits rencontrés avec certains frameworks et d'une utilisation d'outils maintenant obsolètes.

6. Résonateur par guide d'onde de second ordre normalisé. Pour plus d'informations, voir : https://ccrma.stanford.edu/~jos/pasp/Power_Normalized_Waveguide_Filters.html

7. Attention, la Web Audio API ne gère pas la dénormalisation comme nous pourrions nous y attendre. Plus les valeurs d'amplitude sont faibles, plus le CPU est sollicité pour calculer les valeurs flottantes, et nous pouvions même « geler » la page avec un CPU à 100 % pour des niveaux bien au-delà du seuil minimum audible (dans un environnement normalement et sainement amplifié) ? il suffisait de jouer une note et d'attendre le silence pour ne plus pouvoir utiliser l'application ! Nous avons dû gérer en interne ce problème, dans chaque DSP Faust, grâce à la fonction FTZ (désormais présente dans les librairies) implémentée par Stéphane Letz.

8. http://faust.grame.fr/faustplayground/

9. http://michaelbylstra.com/faust-instant/

10. https://blokdust.com/

11. http://www.g200kg.com/en/docs/webmodular/

Pour citer ce document:
Thomas Cipierre, « Le Web, une solution ubiquitaire et accessible pour l'enseignement et la pratique temps-réel de la musique », *RFIM* [En ligne], Numéros, n° 6 - Techniques et méthodes innovantes pour l'enseignement de la musique et du traitement de signal, Mis à jour le 18/06/2018
URL: http://revues.mshparisnord.org/rfim/index.php?id=552