RAPPORT FEEVER

Thomas Cipierre

En tant que stagiaire de master

du 05/05/2014 au 25/07/2014

Les travaux de stagiaire sur les implémentations Faust ont été principalement consacrés à l'utilisation des primitives `rdtable` et `rwtable` de Faust, afin de mettre en place un système de sampling avec variation des vitesses de lecture, antialiasée par interpolation cubique (polynôme du troisième degré sur une fenêtre glissante de 4 valeurs d'amplitude). Les premières implémentations naïves ont permis de faire remonter quelques bugs de compilation utiles aux équipes chargées de la syntaxe du langage.

En tant que chargé de recherche

du 13/10/2014 au 12/04/2015

La mission principale de chargé de recherche était d'implémenter un clavecin synthétisé par modèles physiques[1] en frontend sur le Web[2]. Les optimisations de compilation Faust vers asm.js[3] et la mise en place des normes W3C[4] autour de la WebAudioAPI[5] ont permis d'établir une preuve de concept pour du calcul temps-réel complexe côté client. En effet, les modèles physiques polyphoniques plectre-cordes et une implémentation par modèle de résonance sur 59 filtres passe-bas de deuxième ordre ont permis de proposer aux utilisateurs un réglage temps-réel des paramètres internes du clavecin, ainsi que différents résonateurs (d'une table d'harmonie de clavecin, à une simple réverbération en passant par des modèles de piano, cloche et grand gong) et réglages de tempérament. Les paramètres étaient enregistrables sous forme de presets, et les paramètres avancés activables par des panneaux dédiés – qui, si inactif, reprenaient les valeurs par défaut. Plusieurs pistes MIDI, enregistrées par le claveciniste Martial Morand, étaient également accessibles en lecture.

1 Cipierre, Thomas, Pottier, Laurent, 2015 : « Développement multiplateforme et temps-réel d'un clavecin synthétisé par modèles physiques avec Faust », Proceedings of the Journées d'Informatique Musicale (JIM2015), Montréal, Canada.
2 http://musinf.univ-st-etienne.fr/recherches/ClavecinHtml/web-harpsichord.html
3 sous-typage de bas niveau optimisant du code JavaScript, développé par Mozilla : http://asmjs.org/
4 https://www.w3.org/
5 http://www.w3.org/TR/webaudio/

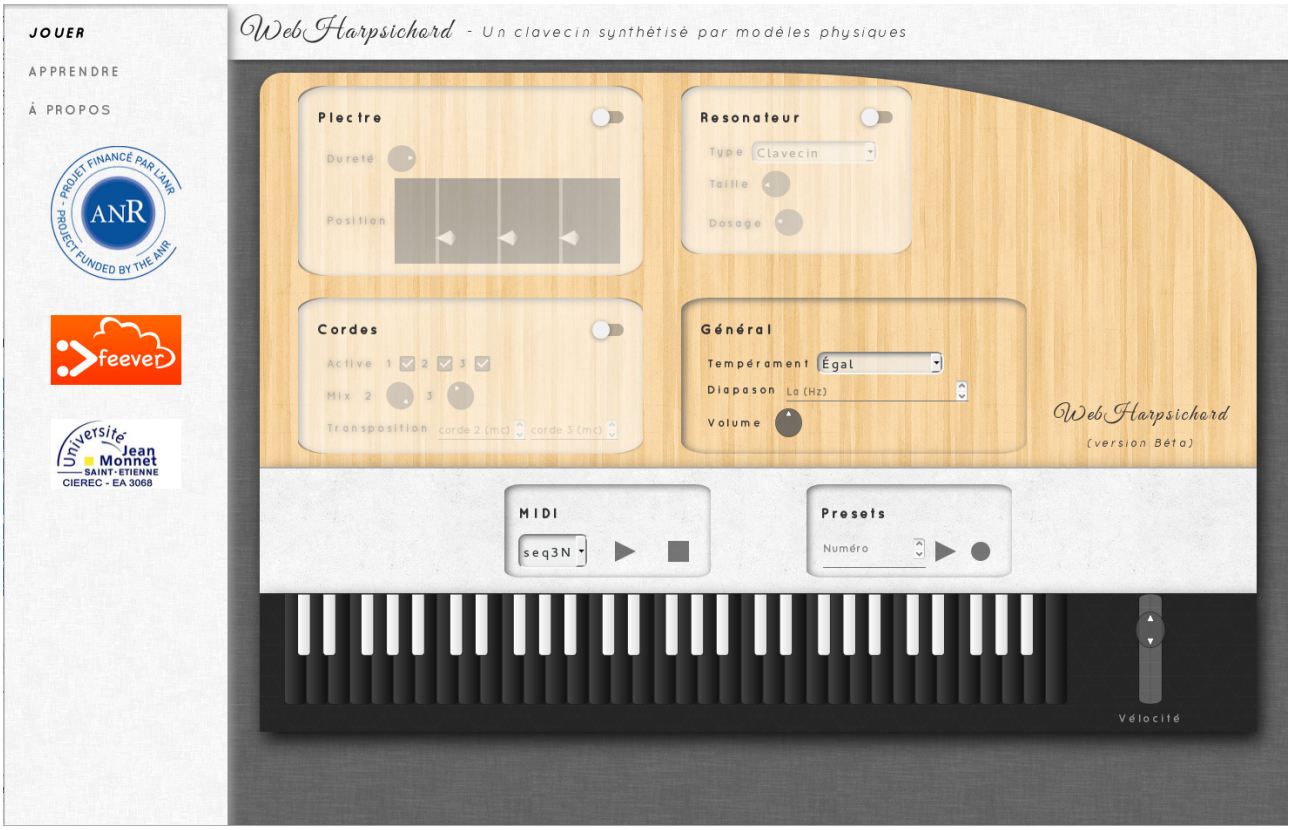

Illustration 1: WebHarpsichord GUI

La plupart des applications Web utilisent les nœuds natifs de la WebAudioAPI, qui montrent plusieurs problèmes d'implémentations[6]. Faust a su innover et rendre possible de nouveaux calculs temps-réel sur le Web jusqu'alors irréalisables, en créant ses propres nœuds WebAudio optimisés par son backend asm.js (et aujourd'hui WebAssembly[7]), tout en gérant la dénormalisation[8] qui n'est pas nativement prise en compte par la WebAudioAPI.

Ce développement concret d'une application avancée a permis de nombreux retours auprès des équipes chargées de la syntaxe de Faust, afin d'obtenir des optimisations de compilations, la gestion de la polyphonie en asm.js, ainsi que la lecture de piste MIDI. Elle a aussi permis de faire remonter plusieurs bugs auprès des développeurs de frameworks Web.

Cependant, les technologies encore très jeunes ont pausé de nombreux soucis d'utilisation – le web n'étant pas encore ubiquitaire face aux implémentations inégales des normes W3C par les divers navigateurs – et demanderaient aujourd'hui une refonte totale pour une utilisation en production ou en milieu éducatif. En effet, Polymer[9] (framework WebComponent[10] développé par Google), alors en version 0.5, se comportait comme convenu sur Chrome, mais posait de gros problème de latence sur Firefox (par l'utilisation de Polyfill – code JavaScript supplémentaire émulant les nouvelles fonctionnalités non implémentées). À l'inverse, asm.js permettait une synthèse idéale sur Firefox,

6 Lazzarini V., Yi, S., Timoney, J., 2015 : « Web Audio : Some Critical Considerations », *Proceedings of the VI Ubimus 2015*, Linnaeus University, Växjö, Sweden.
7 http://webassembly.org/
8 La dénormalisation permet d'éviter des calculs de flottants, de plus en plus coûteux qu'ils augmentent en précision, pour des valeurs d'amplitudes inaudibles.
9 https://www.polymer-project.org/
10 https://www.webcomponents.org/

alors que sujette à de trop nombreux artefacts numériques bloquants sur Chrome. Safari et Internet Explorer, quant à eux, ne produisait aucun son. De plus, seul Chrome gérait nativement le contrôle par clavier MIDI (en version bêta), et l'ajout d'un Shim (code JavaScript supplémentaire émulant les nouvelles fonctionnalités non implémentées) pour Firefox était impossible suite à des conflits avec le Polyfill Webcomponent[11].

En tant que doctorant
du 01/10/2015 au 01/10/2018

La thèse « Développement d'interfaces pour le Web destinées à rendre accessibles les outils de la création musicale contemporaine », financée par la région Rhône-Alpes (ARC 5), a permis de chercher des applications pédagogiques concrètes, à travers le développement d'outils open-source Faust pour le traitement de signal et la synthèse sonore.

Une librairie de synthèse additive (optimisée par lookup-table) a été développée, afin de proposer une interface d'étude sur la génération des formes d'ondes courantes.

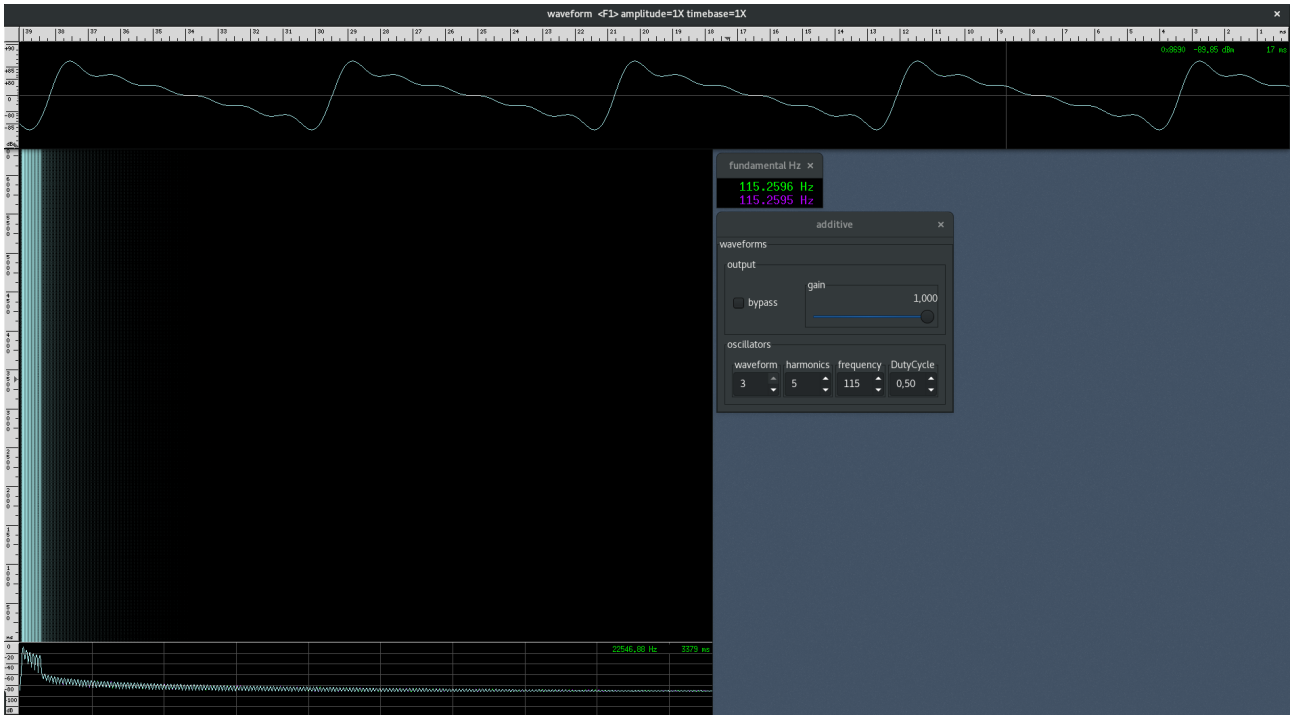

Illustration 2: Synthèse additive

Un environnement de synthèse modulaire a également été développé, afin de proposer des outils de génération de bruit, de synthèse soustractive (filtrage de formes d'onde complexe), de synthèse FM (modulation de la fréquence d'un oscillateur par un autre), de synthèse par synchronisation (réinitialisation de la phase d'un oscillateur par un autre), et des outils de modulation des paramètres de contrôle par LFO (oscillateur basse fréquence), sample-and-hold (envoi de paramètre par mesure discrète d'un signal) et générateur d'enveloppe. La plupart des paramètres sont modulables par tout signal routé dans les entrées prévues à cet effet (avec aliasing). Une vaste documentation est en

11 https://github.com/cwilso/WebMIDIAPIShim/issues/57

cours de rédaction, afin de proposer une approche pédagogique accessible sur le traitement de signal usuel des musiques numériques, illustrée par des exemples en Faust.

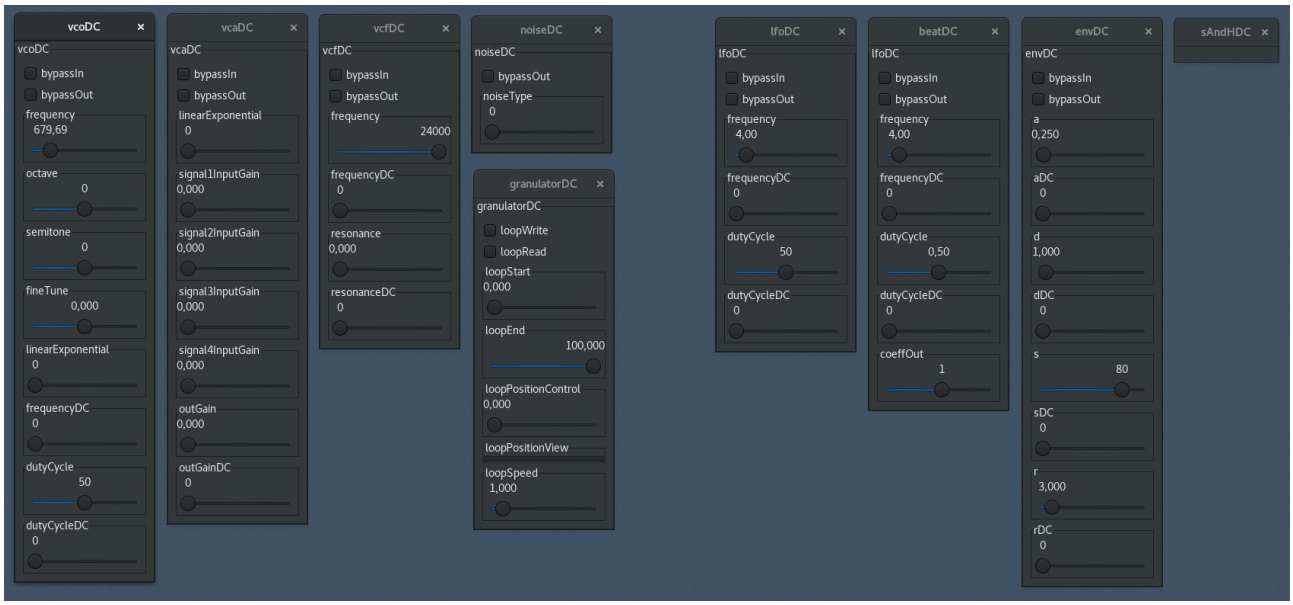

Illustration 3: Synthèse Modulaire

D'autres développement sont en cours, afin de proposer une interface d'utilisation ubiquitaire et ludique, permettant de travailler empiriquement autour du traitement du signal et de la production audionumérique temps-réel sur le Web, en frontend. Grâce à un premier prototypage sur Pure Data, plusieurs tests pédagogiques ont eu lieu dans des classes de licence de Musicologie et d'Arts-Plastiques, et ont abouti sur des restitutions publiques des divers travaux. Une collaboration pédagogique est également en cours avec GRAME et TUMO[12], afin d'intégrer le traitement de signal et la programmation fonctionnelle dans les cursus des adolescents.

Finalement, une contribution à l'intégration de la syntaxe Faust dans Atom[13] a permis de faciliter les approches pédagogiques et le développement des applications, notamment grâce à la refonte de la gestion des librairies Faust par Romain Michon.

12 École arménienne innovant dans la pédagogie et les nouvelles technologies : https://www.tumo.org/
13 Éditeur de texte développé par GitHub : https://atom.io/

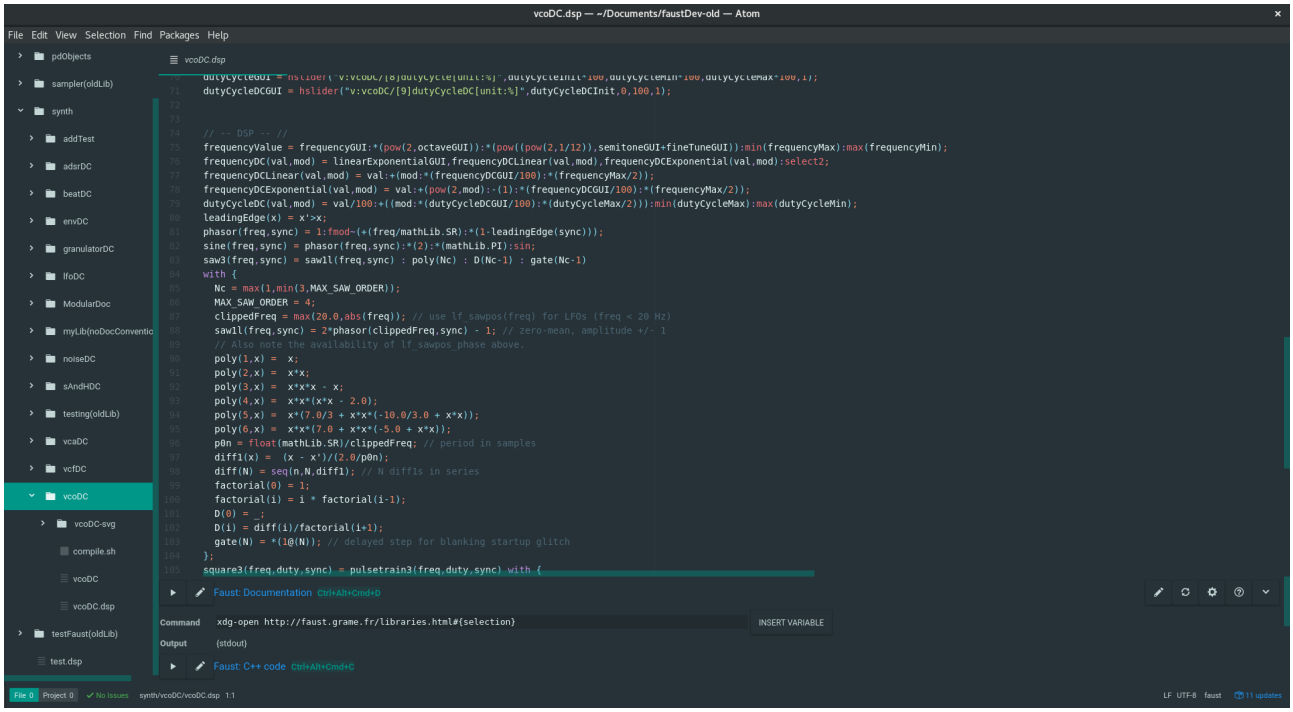

Illustration 4: Atom Faust