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

« Quatre années de workshop Faust »

Romain Michon

Résumé

Faust est un langage de haut niveau pour le traitement du signal audio qui se compile en C++ et permet d'obtenir des plugins et des applications autonomes. Nous présentons les ateliers Faust du CCRMA qui ont été donnés à l'université Stanford et à l'étranger ces dernières années. L'objectif est de présenter en cinq jours un ensemble d'algorithmes simples de traitement du signal. À la fin des ateliers, les participants doivent savoir comment construire des plug-ins audio comme un compresseur, un vocodeur ou une réverbération. Le public rencontré est très éclectique, ingénieurs, informaticiens, hommes d'affaires, musiciens, compositeurs professionnels ou amateurs, mais Faust rend cette tâche, qui serait très difficile avec d'autres plateformes, relativement facile.

Introduction

Depuis une trentaine d'années, le Center for Computer Research in Music and Acoustics (1) (CCRMA) de l'université Stanford (USA) propose chaque été des workshops sur des sujets spécialisés allant du codage audio perceptuel aux musiques procédurales pour les jeux vidéo, etc. (voir le site des workshops d'été du CCRMA (2) pour une liste complète). Pendant cette période, les participants ont un libre accès aux différentes ressources du CCRMA (studios, laboratoire de prototypage, installations informatiques, instruments de musique, etc.). Le prix de l'inscription varie d'un workshop à l'autre entre 450 $ et 650 $.

Les thèmes des workshops donnent généralement un bon aperçu des activités de recherche menées au CCRMA pendant le reste de l'année. Cependant, alors que Dave P. Berners et Jonathan S. Abel avaient pour habitude de proposer chaque été un workshop intitulé *Signal Processing Techniques for Digital Audio Effects* (3), ils ont cessé de le faire en 2011, laissant un vide dans le programme des workshops. Fort de ce constat, nous avons décidé de créer le workshop *Audio Plug-Ins Design With Faust* (4) en 2013. L'idée était de le rendre moins technique et davantage orienté vers les mathématiques que son prédécesseur, visant un public plus large, allant de différents types d'ingénieurs à des artistes.

Depuis une dizaine d'années, au CCRMA, un intérêt croissant est porté au langage de programmation `Faust` (Orlarey *et al.*, 2009) (5) où il est utilisé au coeur de différents projets ainsi que pour l'enseignement. De plus, plusieurs contributeurs du langage font partie des étudiants et du corps enseignant du CCRMA.

Les algorithmes de traitement du signal numérique (DSP (6)) fonctionnant dans le domaine temporel peuvent être exprimés d'une manière très concise et lisible en `Faust`. Aussi, son système d'architecture lui permet de générer un grand nombre de types d'objets (plug-ins, programmes complets, applications web, etc.) compatibles avec différentes plateformes (*OSX*, *Windows*, *Linux*, *Web*, *BELA*, etc.), simplifiant la création de plug-ins audio. Ces raisons nous ont convaincus d'utiliser ce langage pour notre workshop.

Ce chapitre montre comment `Faust` a été utilisé dans le cadre de ce workshop et ce qui en fait un outil privilégié et innovant pour l'enseignement du traitement du signal. Après avoir décrit le format global du

workshop et présenté les différents outils et plateformes qui lui sont liés, nous montrerons comment il a évolué au cours du temps et nous l'évaluerons.

1. Présentation des workshops d'été du CCRMA

Notre workshop sur le *Design de plug-ins audio avec Faust* a été proposé au CCRMA chaque été depuis 2014 (prix de l'inscription : 450 $). Lors de ce cours intensif de cinq jours (à raison d'environ sept heures par jour), les étudiants apprennent les bases du traitement numérique du signal pour l'audio et créent une quinzaine d'effets en partant de rien dans le langage `Faust`. Bien qu'il soit attendu des participants qu'ils aient des notions en informatique musicale et en programmation de manière générale, aucune connaissance en DSP ou en conception de plug-ins audio n'est requise, ce qui nous permet d'accepter des candidats avec des profils très variés.

Grâce à la polyvalence, l'efficacité et la simplicité de `Faust`, la plupart du temps est consacré à l'étude des algorithmes, laissant le choix du « produit fini » pour chaque plug-in aux étudiants. Il est attendu des participants qu'ils obtiennent une bonne maîtrise du langage ainsi qu'un niveau raisonnable de compréhension de notions basiques d'acoustique et des technologies de l'audio numérique d'ici la fin du workshop. Ils doivent également avoir compris et implémenté les effets et filtres suivants : tremolo, modulation en anneau, delay, flanger, écho, distorsion, vocodeur, compresseur, réverbération de type « Schröder » basique, filtres passe-bas et passe-haut simples, filtres en peigne, et banque de filtres.

À cause de certaines limitations de `Faust`, seuls les algorithmes fonctionnant dans le domaine temporel sont étudiés. Ils sont également généralement plus simples à comprendre de manière intuitive sans l'utilisation des mathématiques que les algorithmes fonctionnant dans le domaine fréquentiel.

En octobre 2014, nous avons été invités à donner le workshop *Audio Plug-Ins Design With Faust* à l'Universidad VERITAS de San Jose (Costa Rica) dans le cadre du *Festival de Audio y Acústica Costarricense* (voir la figure 2) par l'un des participants de l'édition de l'été 2013. Le format du workshop était le même que celui donné au CCRMA. Le festival a réuni environ cent cinquante personnes venant de différents pays d'Amérique Centrale. Huit d'entre eux ont participé à notre workshop.

Figure . Le workshop *Design de plug-ins audio avec Faust* (2015)

Figure . Le workshop *Design de plug-ins audio avec Faust* au Costa Rica (2014)



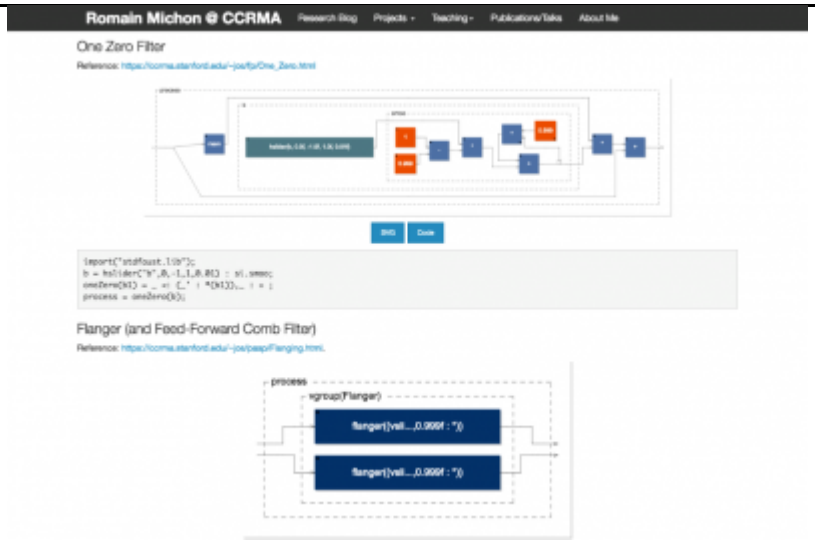

2. Cadre et Support

Tandis que le compilateur de `Faust`, combiné avec `FaustWorks` [(7)] étaient utilisés jusqu'à l'édition 2014 du workshop, `FaustLive` (Denoux *et al.*, 2014) a servi de base pour les années suivantes. Grâce à sa fonction d'« export » (Denoux *et al.*, 2015), `FaustLive` permet aux utilisateurs d'avoir accès à l'ensemble des architectures de `Faust` sans avoir besoin d'installer les SDKs et librairies correspondantes sur leur système. Cela simplifie grandement la préparation des étudiants pour le workshop.

La compilation de code `Faust` « à la volée » dans `FaustLive` a simplifié les processus de prototypage et de débogage pour les participants. Enfin, grâce à `Jack` [(8)], `FaustLive` peut être facilement intégré par les étudiants à leur DAW [(9)] sans avoir besoin de compiler un plug-in.

Des pages wiki [(10)] ont été créées sur le Wiki du CCRMA pour la première édition de notre workshop afin d'y poster les différents codes et algorithmes étudiés. Cette page était mise à jour à la fin de chaque journée. Les années suivantes, elle fut remplacée par une page web afin d'y placer des contenus plus interactifs (voir la figure 3) [(11)]. Par exemple, le schéma fonctionnel interactif de chaque code `Faust` peut être visualisé et « exploré » directement sur la page et les applications web correspondantes générées grâce à l'architecture `Faust asmjs` (Letz *et al.*, 2015) peuvent être utilisées. Les différents wikis et pages web sont toujours disponibles en ligne et peuvent être utilisés par les personnes n'ayant pas pu participer au workshop.

Figure . Page web de l'édition 2016 du workshop *Audio plug-ins design with Faust*



La communication entre les participants et les instructeurs est assurée par une « mailing-list » hébergée sur le serveur du CCRMA.

À cause du profil de certains étudiants, les sujets impliquant beaucoup de mathématiques sont évités autant que possible. Notre but est de donner aux participants une compréhension intuitive des différents algorithmes. Cela n'empêche pas l'utilisation des deux premiers volumes (Smith, 2007a ; Smith, 2007b) de la série d'ouvrage de Julius O. Smith sur le traitement numérique du signal appliqué à l'audio et à la musique comme support de cours. Les différents algorithmes étudiés contiennent tous des liens sur la page web du workshop vers les sections correspondantes de cette série. Nous pensons qu'elle constitue par la suite un point de départ pour les étudiants désireux de parfaire leurs connaissances une fois le workshop terminé.

3. Autres formats de workshops

Mis à part notre workshop *Audio plug-ins design with Faust*, un atelier gratuit d'une journée appelé *Faust Day* est proposé chaque année au CCRMA au mois de janvier. Il s'adresse aux étudiants en Master et en Doctorat, aux professeurs, ainsi qu'aux ingénieurs avec des connaissances approfondies en traitement numérique du signal et en programmation. Une présentation étendue des possibilités de `Faust` est donnée pendant les deux premiers tiers de la journée, et le temps restant est consacré à faire des exercices. L'accent est mis sur l'utilisation du code `C++` généré par le compilateur de `Faust`.

Faust Day a été organisé pour la première fois en 2013 afin de répondre à la demande de certains étudiants du CCRMA désireux d'apprendre le langage `Faust`. Le nombre de participants à *Faust Day* a continué de grandir jusqu'en 2015, année où la fréquentation a été à son maximum (voir la figure 5) avec quarante-et-un participants. Chaque année, plus de la moitié d'entre eux sont des ingénieurs de la Silicon Valley, ce qui montre l'intérêt de l'industrie pour `Faust`.

Une grande partie du budget de ces évènements est allouée à la nourriture et aux boissons afin de renforcer la communauté locale autour de `Faust`.

4. Promouvoir la diversité

En 2015, après deux années de workshop *Audio Plug-Ins Design with Faust*, nous avons remarqué que la plupart des participants étaient des hommes de type caucasien (voir la figure 4). Malheureusement, ce problème est récurrent dans le domaine de l'informatique musicale (Nettingsmeier, 2014) et une tendance similaire a été observée dans d'autre workshop d'été du CCRMA, bien avant que notre workshop ne soit créé. Ce problème a été abordé et traité pour la première fois en 2015 par Fernando Lopez-Lezcano et

Bruno Ruviaro dans leur workshop d'été sur *SuperCollider*, en créant une bourse intitulée *Women in Computer Music* [(12)] prenant en charge les frais d'inscription des lauréates. Inspirés par cette initiative, nous avons décidé de l'appliquer à notre workshop [(13)]. Quatre femmes parmi les dix-sept candidates se sont vues attribuer la bourse en 2016, ce qui eut un fort impact sur le profil démographique des participants du workshop (voir la section « Évaluation »). De plus, cela a permis d'attirer l'attention de plus de femmes sur notre workshop, et certaines des candidates ont décidé d'y participer sans l'aide de la bourse.

5. Cours en ligne

L'édition 2015 du workshop *Audio Plug-Ins Design with Faust* a été entièrement filmée et transformée en un cours en ligne gratuit [(14)]. Les différents codes `Faust` peuvent être compilés directement dans le navigateur web de manière similaire que sur la page web du workshop (voir la section « cadre et support »), rendant l'apprentissage plus interactif. Un total de trente-quatre heures de vidéo a été enregistré et mis en ligne sur *YouTube*.

6. Autres utilisations de Faust pour l'enseignement au CCRMA

`Faust` a été utilisé au CCRMA dans le cadre de plusieurs classes depuis une dizaine d'années. Julius O. Smith a intégré, à partir de 2017, des codes `Faust` en parallèle de ses exemples en `Matlab` et `C++` pour sa série de cours sur *Introduction to Audio Signal Processing* [(15)] (*Music 320a/b*) [(16)]. De façon similaire, il a ajouté des exemples `Faust` à son cours sur *Signal Processing Models in Musical Acoustics* [(17)] (*Music 42*

De manière plus récente, `Faust` a été utilisé par des étudiants dans le cadre de leur projet de fin de trimestre pour les cours de *Music, Computing, and Design* [(19)] (*Music 256a/b*) [(20)] ainsi que dans la série de *Fundamentals of Computer-Generated Sound* [(21)](*Music 220a*) [(22)].

Enfin, `Faust` sera intégré à partir de 2016 au programme de *Music 220a* et *Music 256a* pour être au coeur de ces deux classes.

7. Evaluation

7.1. Le langage Faust

En sa qualité de langage de programmation fonctionnel, `Faust` est un outil assez inhabituel pour l'enseignement du traitement du signal. Dans la mesure où son paradigme est très différent de celui de `C++` et `Matlab`, il est souvent plus accessible pour les personnes avec peu d'expérience en programmation que pour des programmeurs confirmés. Le fait que `Faust` n'a pas de typage et que tout peut être considéré comme un signal le rend très intuitif pour des ingénieurs électrotechniques ainsi que pour les programmeurs débutants. De manière similaire, le système d'architecture de `Faust`, combiné à sa sémantique simplifiée, permet aux étudiants de se concentrer sur l'implémentation des algorithmes sans avoir à se soucier du produit final. Grâce à leur aspect graphique, les opérateurs de `Faust` (ex. `: <:, :>` , etc.) sont faciles à interpréter. Enfin, `Faust` a relativement peu de bugs, ce qui simplifie grandement son apprentissage.

Toutefois, certains concepts restent moins abordables que d'autres, notamment l'utilisation de l'opérateur tilde (~) lors de la création de signaux récursifs. De plus, dans la mesure où `Faust` est continu, l'implémentation de machines d'états peut être compliquée. Enfin, le fait que `Faust` soit « uni-horloge » limite la gamme d'algorithmes pouvant être implémentés avec ce langage.

8. Le compilateur de Faust et son environnement

Comme cela a été mentionné dans la section précédente, le compilateur de `Faust` a relativement peu de bugs, ce qui est très important pour les débutants qui partent toujours du principe qu'ils sont responsables des problèmes qu'ils rencontrent. Le fait de pouvoir visualiser le schéma fonctionnel d'un code `Faust` est très utile pour l'apprentissage de ce langage, en particulier lors de l'utilisation de l'opérateur tilde. La version du compilateur de `Faust` basée sur LLVM [(23)] intégrée à `FaustLive` permet de compiler un code `Faust` en un exécutable de manière presque instantanée. Cela permet de gagner beaucoup de temps lors du prototypage et du développement.

La principale limitation du compilateur de `Faust` est son système de débogage. Les messages d'erreurs peuvent parfois être difficiles à interpréter (les numéros de ligne peuvent être incorrects) et ne correspondent pas toujours à l'erreur concernée, ce qui peut être très problématique, notamment pour les débutants. De façon similaire, il manque à `Faust` une série d'outils de débogage DSP pour tracer les signaux sous différentes formes, effectuer des analyses fréquentielles sur ces derniers (ex. FFT), etc.

9. Évolution du profil démographique des participants

Les workshops *Audio Plug-Ins Design with Faust* et *Faust Day* attirent chaque année des personnes aux profils très variés allant du retraité à différents types d'ingénieurs (voir les figures 4 et 5). Bien que ceci rende l'enseignement plus difficile, cela n'a jamais été perçu comme une limite pour nous. Dans la mesure où le contenu du workshop est très dense, nous savons que les participants venant avec peu de connaissances en traitement du signal apprennent moins vite que certains ingénieurs, mais nous faisons en sorte qu'ils maîtrisent les points principaux.

Figure . Évolution du profil des participants du workshop *Audio Plug-Ins Design with Faust* entre 2014 et 2016

	2014	Costa Rica 2014	2015	2016
Ingénieurs	30 %	40 %	30 %	0 %
Lycéens	8 %	0 %	0 %	0 %
Étudiants en licence	15 %	0 %	7 %	0 %
Étudiants en master ou doctorat	8 %	40 %	14 %	13 %
Professeur	15 %	0 %	7 %	0 %
Compositeur	8 %	0 %	14 %	50 %
Retraité	8 %	0 %	14 %	12 %
Autre	8 %	20 %	14 %	25 %
Femmes/Hommes	0 %	0 %	14 %	63 %
TOTAL	13	5	14	8

Figure . Évolution du profil des participants du CCRMA *Faust Day* entre 2013 et 2016

	2013	2014	2015	2016
Ingénieurs	61 %	48 %	59 %	55 %
Étudiants	28 %	44 %	12 %	22 %
Compositeurs	0 %	4 %	3 %	6 %
Autre	11 %	4 %	12 %	17 %
Femmes/Hommes	6 %	4 %	8 %	12 %
TOTAL	18	23	41	36

En offrant quatre bourses *Women in Computer Music* lors de l'édition 2016 de notre workshop, nous avons complètement changé le profil démographique des participants en obtenant un équilibre entre le nombre d'hommes et de femmes. De plus, les femmes ayant participé à l'édition 2016 du workshop étaient presque toutes de groupes ethniques différents.

La popularité des *Faust Days* a crû de manière rapide (voir la figure 5) et nous n'avons plus besoin de les promouvoir. Ils constituent une manière efficace de faire connaître `Faust` dans la Silicon Valley et ils permettent de renforcer la communauté locale de développeurs et d'utilisateurs. Chaque année, des ingénieurs de grandes entreprises dans le domaine de l'audio tel que *Smule*, *Dolby*, *Korg*, *McDSP*, *Apple*, *Google*, etc. participent à ces évènements. La diversité est un problème majeur lors des *Faust Days*, mais prendre des mesures pro-actives telles que la création d'une bourse, comme cela a été fait pour notre workshop, n'est pas adapté dans la mesure où les *Faust Days* sont gratuits et ne sont pas limités en termes de places.

De manière générale, nous pensons que les participants continuent d'utiliser `Faust` après le workshop dans la mesure où ils nous envoient régulièrement des questions par la suite. De plus, certains d'entre eux sont devenus des contributeurs actifs du langage.

Conclusion

Après trois années de workshop *Audio Plug-Ins Design with Faust*, `Faust` s'est avéré être un outil extrêmement fiable pour l'enseignement du traitement numérique du signal. Il nous a permis de nous concentrer sur les algorithmes en minimisant le temps passé sur l'implémentation des plug-ins.

Le paradigme de programmation de `Faust` aide à minimiser les disparités entre les programmeurs expérimentés et les débutants, rendant l'enseignement plus simple dans un contexte où les étudiants ont des profils très différents. Nous pensons que `Faust` est un langage parfaitement adapté à l'enseignement du traitement numérique du signal, c'est pourquoi nous continuerons de l'utiliser pour les futures éditions de notre workshop.

Pour la suite, nous espérons fournir un meilleur support en ligne pour l'apprentissage du DSP via `Faust` et améliorer la documentation du langage. Créer une plateforme en ligne pour l'échange de code `Faust` serait également très bénéfique pour la communauté d'informatique musicale.

---

1. Centre pour la recherche informatique en musique et en acoustique. https://ccrma.stanford.edu: Site web du CCRMA (lien vérifié le 23 août 2017).

2. https://ccrma.stanford.edu/workshops : Site web des workshops d'été du CCRMA (lien vérifié le 23 août 2017).

3. Techniques de traitement du signal pour les effets audionumériques.

4. Design de plug-ins audio avec Faust.

5. http://faust.grame.fr : Site web de Faust (lien vérifié le 23 août 2017).

6. Digital Signal Processing.

7. http://www.grame.fr/logiciels/faustworks : Page web de FaustWorks (lien vérifié le 23 août 2017).

8. http://www.jackaudio.org/ : Site web de Jack Audio (lien vérifié le 23 août 2017).

9. Digital Audio Workstation.

10. https://ccrma.stanford.edu/wiki/FaustWorkshop2014 : Page wiki de l'édition 2014 du workshop « Audio Plug-Ins Design with Faust » (lien vérifié le 23 août 2017) ; https://ccrma.stanford.edu/wiki/FaustWorkshop2014-CostaRica : Page wiki de l'édition Costaricaine du workshop « Audio Plug-Ins Design with Faust » (lien vérifié le 23 août 2017).

11. https://ccrma.stanford.edu/~rmichon/faustWorkshops/2015 : Page web de l'édition 2015 wu workshop « Audio Plug-Ins Design with Faust » (lien vérifié le 23 août 2017) ; https://ccrma.stanford.edu/~rmichon/faustWorkshops/2016 : Page web de l'édition 2015 wu workshop « Audio Plug-Ins Design with Faust » (lien vérifié le 23 août 2017).

12. Femmes dans l'informatique musicale.

13. https://ccrma.stanford.edu/women-in-computer-music-audio-plug-ins-designed-with-faust-workshop-scholarship Page we

14. https://ccrma.stanford.edu/~rmichon/faustWorkshops/course2015 - Cours de Faust en ligne du CCRMA (lien vérifié le 23 août 2017).

15. Introduction au traitement numérique du signal pour l'audio.

16. https://ccrma.stanford.edu/courses/320/ - Page web de la série de cours Music 320 du CCRMA (lien vérifié le 23 août 2017).

17. Modèles de traitement du signal pour l'acoustique musicale.

18. https://ccrma.stanford.edu/courses/420 - Page web du course Music 420 du CCRMA (lien vérifié le 23 août 2017).

19. Musique, informatique et design.

20. https://ccrma.stanford.edu/courses/256a - Page web du course Music 256a du CCRMA (lien vérifié le 23 août 2017).

21. Fondamentaux des sons générés par ordinateur.

22. https://ccrma.stanford.edu/courses/220a - Page web du cours Music 220a au CCRMA (lien vérifié le 23 août 2017).

23. Low Level Virtual Machine : http://llvm.org (lien vérifié le 23 août 2017).

Pour citer ce document:
Romain Michon, « Quatre années de workshop Faust », *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 14/06/2018
URL: http://revues.mshparisnord.org/rfim/index.php?id=538