Camomile, enjeux et développements d'un plugiciel audio embarquant Pure Data

Pierre Guillot

▶ **To cite this version:**

Pierre Guillot. Camomile, enjeux et développements d'un plugiciel audio embarquant Pure Data. Journées d'Informatique Musicale (JIM 2018), May 2018, Amiens, France. hal-01791392

HAL Id: hal-01791392
https://hal.science/hal-01791392v1

Submitted on 14 May 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

CAMOMILE, ENJEUX ET DEVELOPPEMENTS D'UN PLUGICIEL AUDIO EMBARQUANT PURE DATA

Pierre Guillot
CICM – EA1572
Université Paris 8
guillotpierre6@gmail.com

RÉSUMÉ

Cet article présente Camomile un outil plugiciel audio multiplateforme, compatible avec de multiples formats et permettant de charger des patchs Pure Data dans les stations de travail audionumérique. Cette présentation reviendra tout d'abord sur les contextes d'utilisation dans lesquels la nécessité d'un tel outil est apparue. L'objectif sera de mettre en avant les principaux enjeux du projet sur un plan fonctionnel, usuel et technique. Par la suite, un état de l'art, général mais non exhaustif, sera présenté en revenant sur les différentes propositions et les évolutions dans ce domaine. Parcourir ces travaux, des premières approches des logiciels de type *patcher* embarqués jusqu'aux projets plus récents de plugiciels audio, permettra notamment de clarifier les difficultés, les contraintes et les questions soulevées par cette démarche afin de comprendre les choix opérés lors de la mise en œuvre du projet Camomile. Des solutions seront alors apportées en revenant sur les principaux axes du développement de l'outil. Il s'agira de présenter les choix importants opérés au cours des différentes versions et les répercussions sur l'outil final et son usage. Enfin, un bilan et les perspectives de ce projet seront exposés.

1. INTRODUCTION

Camomile[1] est un outil permettant de créer des plugiciels audionumériques – des modules externes destinés à être utilisés dans les stations de travail audionumériques – à partir de patchs Pure Data [1][2]. Cet outil est lui-même un plugiciel aux formats VST2, VST3[3] et Audio Unit[4] pour les systèmes d'exploitation Linux, Windows et MacOs embarquant le moteur de Pure Data. Après avoir créé un traitement ou un synthétiseur sonore dans Pure Data, l'utilisateur peut utiliser les plugiciels de Camomile comme base, et leurs associer les patchs élaborés afin de concevoir des plugiciels autonomes (Figure 1).

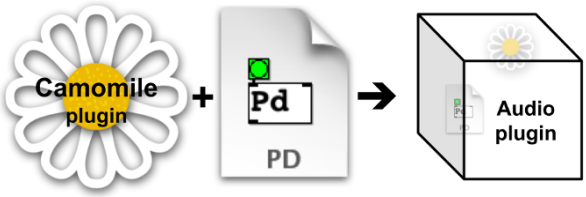

Figure 1. Fonctionnement de la création d'un nouveau plugiciel à partir d'un patch Pure Data et d'un plugiciel Camomile.

Cette approche offre donc à la fois les avantages du plugiciel, notamment via la variété des stations de travail audionumérique à laquelle elle donne accès, ainsi que ceux de Pure Data, à savoir son dynamisme et sa modularité. De la part de l'utilisateur, il s'agira de comprendre comment créer des patchs afin de tirer profit de toutes les fonctionnalités offertes par les plugiciels et les stations de travail. Ce sujet est présenté dans la documentation fournie avec l'outil Camomile et sera traité dans une prochaine publication. Car, avant cela, il semble intéressant de revenir sur les raisons de la mise en œuvre de cet outil ainsi que sur les choix opérés lors de celle-ci. En effet, c'est de cela même que résultent les modalités d'utilisation de l'outil. Cet article revient donc dans la section 2, sur le contexte et les enjeux de cette mise en œuvre, afin de présenter les spécifications techniques fonctionnelles qui ont permis de déterminer les choix de développement. Puis, dans la section 3, les travaux à la fois préexistants et parallèles au projet Camomile seront présentés, afin de mettre en avant les questions et les problématiques importantes qu'il est nécessaire de résoudre. Suite à cela, dans la section 4, les axes importants de la mise en œuvre de Camomile seront explorés, en allant des premiers développements en mai 2015 à la dernière version publiée en décembre 2017 (Figure 2). Enfin dans la section 5, ce travail sera mis en

[1]Les différentes versions du plugiciel sont disponibles en ligne sur le répertoire Github du projet. Le code source est ouvert, libre et gratuit et est disponible sur ce même répertoire. Depuis la version 0.1.0, les sources sont distribuées sous licence GNU GPLv3. Les sources des versions antérieures sont distribuées sous licence BSD 3 github.com/pierreguillot/camomile (site consulté en janvier 2018).

[2]Pure Data est un logiciel de type *patcher* libre et gratuit, créé par Miller Puckette à l'Université de San Diego en Californie msp.ucsd.edu/software.html (site consulté en janvier 2018).

[3]Les formats de plugiciel audionumérique VST (Virtual Studio Technology) 2 et 3 sont développés par la société Steinberg www.steinberg.net (page consultée en janvier 2018).

[4]Le format de plugiciel audionumérique Audio Unit est développé par Apple developer.apple.com/audio/ (page consulté en janvier 2018).

Pierre Guillot, *Camomile, enjeux et développements d'un plugiciel audio embarquant Pure Data*, Journées d'Informatique Musicale (JIM 2018), Amiens, France, éd. L. Bigo, M. Giraud, R. Groult, F. Levé, pp. 15–23, 2018. CC BY-SA 4.0.

perspective en revenant sur les questions qu'il reste à explorer mais aussi sur le potentiel offert par un tel outil.

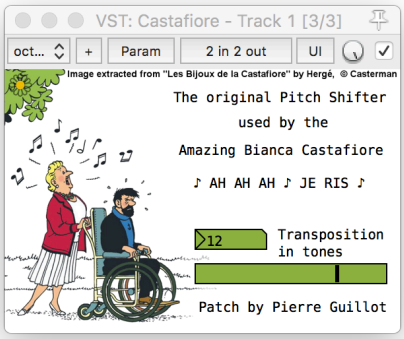

Figure 2. Le plugiciel *Castafiore* distribué en exemple avec la dernière version de Camomile et permettant de modifier la hauteur du signal sonore en fonction d'un paramètre de transposition contrôlable via l'interface graphique avec soit une boîte nombre, soit une glissière (*slider*) horizontale.

2. CONTEXTES ET ENJEUX

Le plugiciel Camomile a été élaboré afin de répondre à l'envie et la nécessité d'utiliser des patchs Pure Data au sein d'une station de travail audionumérique. Bien que relativement spécifique, ce besoin se manifeste dans des pratiques, des usages et des contextes variés, qui répondent à différents enjeux, desquels découlent des spécifications qui peuvent être complémentaires. Cette partie revient sur ces questions, afin de comprendre et mettre en perspective les travaux préexistants présentés par la suite, ainsi que les choix qui ont été opérés lors de la mise en œuvre de Camomile.

2.1. Contextes du projet

En 2015, au début du projet, l'envie d'utiliser des patchs dans une station de travail audionumérique provient du besoin de faciliter le contrôle des traitements audionumériques mis en œuvre dans un contexte de temps différé. L'objectif est notamment de tirer profit de la ligne temporelle généralement présente dans les stations de travail audionumérique[5]. De nombreux praticiens de l'informatique musicale ont d'ailleurs fait part de leurs attentes à ce sujet, ce qui amène notamment en 2013 l'équipe du CICM, dans le cadre des projets HOA [4], à concevoir un plugiciel VST offrant alors la possibilité de spatialiser des sources directionnelles en ambisonie [5][6]. Il a alors été envisagé de proposer d'autres traitements originaux de l'espace et du son – tels que ceux offerts par les versions pour les logiciels Max[7] et Pure Data de la bibliothèque HOA[8] – mais faute de temps et en raison de la complexité des opérations nécessaires à leurs mises en œuvre, cet objectif n'a jamais pu être réalisé. Suite au prolongement des expérimentations et à l'approfondissement des travaux de recherches sur la spatialisation du son, notamment via le logiciel Pure Data [7], la seule solution viable pour répondre à ce problème semble être un plugiciel flexible et dynamique permettant de charger des patchs. Cette approche permet d'assurer une bonne reproduction du rendu sonore des traitements réalisés sous forme de patch au sein des stations de travail audionumérique, tout en limitant le temps de développement. Les logiciels de type *patcher* sont idéals pour l'expérimentation mais prendre en compte et intégrer dans un plugiciel les nombreux et fréquents changements que cela implique est très chronophage. Ainsi, l'enjeu du projet est aussi d'offrir un outil plus adapté à un contexte de développement et recherche. Enfin, la nécessité d'un tel outil s'est fait ressentir dans un cadre pédagogique. Le département de Musique de l'Université Paris 8 propose un cours de 2ème année de Licence d'Introduction à la Programmation avec Max et Pure Data[9] où beaucoup d'étudiants découvrent pour la première fois les logiciels de type *patcher* [9]. Aussi, l'un des défis pédagogiques est de démontrer le potentiel de ces outils via, entre autres, des conditions et des contextes concrets d'utilisations. Les étudiants sont familiers des stations de travail audionumérique et les utilisent généralement déjà fréquemment. Ainsi, proposer un plugiciel permettant d'utiliser un traitement créé sous forme de patch directement dans un de ces logiciels, offre une réponse à ce problème[10]. Ce projet tente donc de répondre à des problématiques provenant d'une approche d'utilisateurs, de développeurs mais aussi d'enseignants.

2.2. Spécifications

Ces observations amènent à définir un certain nombre de spécifications fonctionnelles et techniques.

[5]Dans cet article, il ne s'agit pas d'approfondir la question de la ou des représentations du temps offertes de manière native au sein du logiciel Pure Data, d'autant qu'il est possible de combiner Pure Data à d'autres outils d'écriture temporelle tels que les logiciels Ossia développé au LaBRI [2] ou Iannix [3]. Il s'agit simplement de mettre en avant un besoin et une attente qui est plus liée à des questions d'accessibilité et de facilité de mise en œuvre et d'utilisation.

[6]Le plugiciel est à présent non fonctionnel mais il est cependant toujours disponible sur le site du projet HOA www.mshparisnord.fr/hoalibrary (site consulté en janvier 2018).

[7]Le logiciel Max [6], originellement développé à l'IRCAM par Miller Puckette est aujourd'hui développé et distribué par la société Cycling'74 cycling74.com (site consulté en janvier 2018).

[8]Ces opérations sont par exemple construites autour de la synthèse granulaire ou d'un filtre à réponse impulsionnelle infinie réalisés sous forme de patch dans les logiciels Max et Pure Data. La complexité est alors de recréer en C et C++ l'ensemble des traitements audionumériques des objets utilisés, ainsi que leurs réseaux de connexions mais aussi les interfaces graphiques utilisateurs, même basiques, permettant de contrôler ces traitements.

[9]Ce cours a été dispensé depuis de nombreuses années par Anne Sèdes, Alain Bonardi, puis par moi-même en 2016 et 2017 et à présent par E. Maestri qui intègre notamment le logiciel Kiwi [8] à l'apprentissage.

[10]Au cours de ces années d'enseignement, savoir s'ils pouvaient utiliser leurs patchs sur des stations de travail audionumérique était en effet une question récurrente des étudiants.

2.2.1. Intégration du moteur patcher

Afin de répondre à l'ensemble de ces usages dans des contextes compositionnels, expérimentaux, pédagogiques ou de développements, l'idéal serait évidemment que l'outil fonctionne sur le plus grand nombre de plateformes logicielles – et donc le plus grand nombre de formats de plugiciel audionumérique – et de systèmes d'exploitation. L'objectif serait de concevoir un outil compatible avec les systèmes Linux, MacOs et Windows[11], sous forme de plugiciel intégrant les technologie VST, Audio Unit ou encore LV2[12]. Sur un aspect plus fonctionnel, il est primordial que le rendu sonore du patch chargé au sein de la station de travail audionumérique soit identique à celui offert par l'application de bureau originel qui a permis de le créer. Cette spécification amène à réutiliser le moteur sonore originel mais aussi implicitement à éviter les logiciels au code source fermé, qui empêchent donc toute réutilisation[13]. De manière générale, le plugiciel final devrait satisfaire le maximum de spécifications définies par les différents formats et les stations de travail audionumérique, et notamment la capacité de gérer plusieurs instances, mais aussi d'être utilisable dans un contexte de *multithreads*[14]. Le premier enjeu principal du développement est donc, au travers du plugiciel, de réaliser le lien entre la station de travail audionumérique et le moteur de type *patcher*, en faisant concorder leurs spécifications et leurs recommandations respectives.

2.2.2. Gestion du moteur patcher et du plugiciel

Évidemment, afin d'être utilisable, l'outil doit aussi proposer à l'utilisateur un moyen de définir et contrôler l'ensemble des fonctionnalités génériques d'un plugiciel, telles que les paramètres, les préréglages, la latence, etc. L'une des spécifications à ce sujet serait que l'utilisateur souhaitant créer un patch destiné au plugiciel audio n'ait pas à devoir coder en dehors des langages initiaux de type *patcher*, ou du moins n'ait pas à procéder à des opérations qui demandent des notions complexes d'informatique, telles que la compilation d'un code. En effet, si l'outil final est destiné à des étudiants, ce type d'approche peut rapidement décourager son utilisation. Cette spécification liée aux questions d'accessibilité est aussi importante dans un contexte de développement, car elle assure la portabilité d'un même patch sur l'ensemble des plateformes logicielles et des systèmes d'exploitations supportés par le plugiciel. Enfin, les logiciels de type *patcher* étant des applications de programmation graphique, il serait aussi intéressant de pouvoir offrir la possibilité de créer l'interface graphique utilisateur du plugiciel, via l'interface originale du programme. Le deuxième enjeu principal est de créer un système commun de gestion et de communication entre le logiciel de type *patcher* et le plugiciel. Cela afin de pouvoir définir les propriétés du plugiciel à partir du *patcher* et d'adapter le fonctionnement des patchs en fonction des informations envoyées par le plugiciel.

3. ÉTAT DE L'ART

Avant de proposer une réponse à ces enjeux et ces spécifications en exposant les choix réalisés lors de la mise en œuvre du plugiciel Camomile, cette partie revient sur les propositions préexistantes ou développées parallèlement, afin de s'en inspirer et d'éviter les problèmes qu'ils ont pu rencontrer.

3.1. Des origines de Max à Max for Live

L'idée d'utiliser des patchs en tant que module d'extension au sein d'un système tiers n'est pas nouvelle. Dès 1988, l'un des objectifs du logiciel Max est notamment de pouvoir créer des synthétiseurs audionumériques sous forme de patch, pouvant être utilisés sous forme de carte de traitement du signal[15]. Cette proposition ne restera longtemps qu'à l'état d'idée. En effet, à ses débuts le logiciel était exclusivement orienté sur le traitement de messages et l'intégration du traitement du signal audionumérique dans Max n'apparaîtra qu'en 1991 [10]. C'est à la suite du développement de l'audionumérique avec l'émergence des stations de travail audionumérique telles que Cubase en 1996 du format VST[16], que l'idée pourra voir le jour. Notamment en remplaçant les cartes de traitement du signal par des plugiciels audionumériques. En 1998 la société Cycling'74 propose une extension à Max nommée Pluggo [11], permettant de charger des patchs au sein d'une station de travail audionumérique via un plugiciel VST. Cet outil offre un nombre de fonctionnalités très complètes, telles que la création du moteur sonore, la création d'une interface graphique originale et d'un système permettant de configurer et gérer les paramètres, notamment à l'aide d'objets Max dédiés spécifiquement à cet usage. Le plugiciel de Pluggo fonctionne avec une version d'exécution de Max embarquée[17] qui permet de charger dynamiquement des patchs. Cet outil offre aussi la possibilité de générer des

[11]Il est raisonnable d'affirmer que l'usage de cet outil sur d'autres systèmes d'exploitation ou avec d'autres formats de plugiciel audionumérique se révèle purement anecdotique pour le moment.

[12]Les informations relatives au format LV2 sont disponibles sur le site lv2plug.in (site consulté en janvier 2018).

[13]Cette spécification est d'autant plus en accord avec une approche de recherche et recoupe des questions pédagogiques car les applications libres et ouvertes sont aussi souvent gratuites ou du moins plus accessibles.

[14]Système permettant plusieurs tâches d'exécutions en parallèle, ce qui est souvent le cas sur les ordinateurs modernes et les logiciels actuels.

[15]"*Max has hooks for patching a digital synthesizer, which may just be (as now) an interpreting program in the Macintosh or may (in the future) be a plug-in signal-processing card.*", [9], p. 420.

[16]En 1996, la société Steinberg publie la station de travail audionumérique Cubase 3.02 avec notamment l'intégration de la technologie VST, information d'après artisteaudio.fr/steinberg-cubase/ (page consultée en janvier 2018).

[17]Il est possible exclusivement de jouer le patch. L'édition doit être réalisée au sein du logiciel Max original.

nouveaux plugiciels en embarquant directement les patchs dans des copies indépendantes et distinctes du plugiciel original de la distribution. Le développement du projet prend fin en 2006 avec la version 3.6.1 pour Max 4.6.2, ce qui la rend aujourd'hui dépréciée et restreint fortement toute utilisation. La technologie est cependant réintégrée en 2009 au sein de l'extension Max for Live, permettant de charger des patchs au sein du logiciel Ableton Live[18], dont le fonctionnement y est plutôt similaire[19]. Cet outil puissant, de par ses nombreuses années de développement et d'usage, souffre néanmoins d'une double restriction. Celui-ci n'est plus réservé qu'à une seule station de travail audionumérique, Ableton Live, dont la technologie est fermée tout comme celle du logiciel Max. Aussi, cela freine son appropriation dans un contexte de recherche, où les notions de propriété et d'ouverture sont importantes[20] . De même dans un contexte d'utilisation lié à un cadre pédagogique, où le prix cumulé des technologies devient rapidement un frein à l'appropriation des outils de la part des étudiants universitaires.

3.2. De Pure Data au moteur libpd

Le logiciel Pure Data, par sa gratuité, son fonctionnement multiplateforme, son code source ouvert, est accessible à tout un chacun et accepte les ajouts et modifications par des personnes extérieures[21]. Il est ainsi une excellente alternative à l'usage de Max et offre un terrain pour l'émergence d'outils adaptés aux besoins pédagogiques mais aussi de recherche. C'est donc naturellement vers cet outil que les développeurs se tournent lorsqu'il s'agit de s'approprier un moteur audio offrant un modèle de type *patcher*. Pure Data est déjà utilisé par de nombreuses plateformes et de nombreux outils logiciels en tant que moteur audio. Un des premiers projets utilisant cette technologie est PDa (Pure Data Anywhere), réalisé en 2003 par Günter Geiger [13] et visant à faire fonctionner le moteur audio du logiciel sur un PocketPC. Par la suite en 2010, un ensemble de développeurs[22] proposent libpd, une bibliothèque en C embarquant le moteur[23] de Pure Data et offrant des interfaces de programmation en C++, Java, Processing, Objective-C et Python, qui permettent notamment un support pour les plateformes Android and iOS [14]. Cette bibliothèque est utilisée dans de très nombreux projets[24], dont les applications pour téléphones et tablettes tactiles iOS[25] avec PdParty de Dan Wilcox [15] ou son homologue pour Android, dont elle est inspirée avec PdDroidParty[26], développée par Chris McCormick. Dès lors qu'il est possible d'embarquer le moteur de Pure Data au sein d'autre application via libpd, plusieurs projets de plugiciels permettant de charger des patchs Pure Data voient le jour. Par exemple, le plugiciel au format LV2, PdLV2[27] qui fonctionne sur les systèmes d'exploitation Linux et dont le développement date d'au moins 2013, ou plus récemment le plugiciel VST, PdPulp[28] qui fonctionne sur le système d'exploitation MacOs, et dont le développement est contemporain de celui de Camomile. Ces projets sont extrêmement expérimentaux et possèdent sensiblement moins de fonctionnalités qu'un outil tel que Max for Live. Ils sont par conséquent encore difficilement utilisables dans un contexte de production. Il n'est pas donc nécessaire de revenir avec précision sur ces mises en œuvre, mais il est important de relever un problème crucial lié directement à l'architecture du moteur Pure Data. Le logiciel ayant été pensé comme une application autonome, le noyau du code restreint son usage et le résultat concret pour les utilisateurs est que seule une instance d'un plugiciel peut être chargée dans la station de travail audionumérique[29]. Ce problème est primordial à résoudre car l'utilisation multiple d'un même plugiciel dans les chaînes de traitement audio est ancré dans les usages.

3.3. Au-delà de libpd

Deux approches ont néanmoins réussi si ce n'est à résoudre le problème, du moins à le contourner. L'une se trouve au sein du plugiciel PdVst~ originellement développé en 2004 par Joseph Jarlo[30] puis repris depuis 2016 par Jean Yves Gratius[31]. Dans ce plugiciel VST,

[18]Le transfert de la technologie de Pluggo vers Max for Live a été annoncé en mai 2009 sur le site de Cyling'74 cycling74.com/newsletters/pluggo-technology-moves-to-max-for-live (page consultée en janvier 2018).

[19]Évidemment de nouvelles fonctionnalités sont apparues et/ou ont été modifiées mais le mécanisme général y est relativement similaire.

[20]La comparaison entre le développement de Pure Data et de jMax est un bon exemple de ce problème [12].

[21]"*Pd was instantly embraced by a huge, and extremely hip, community of users who have taken it far beyond my wildest dreams of it [...] One meaning of Pd was "Public Domain"*, [12], p. 200.

[22]Les créateurs de libpd sont à l'époque une équipe rencontrée sur internet et composée de Peter Brinkmann, Peter Kirn, Richard Lawler, Chris McCormick, Martin Roth & Hans-Christopher Steiner.

[23]Le moteur est ici pris dans son sens le large, avec la partie dédiée à l'audio, aux messages, au MIDI et autres normes, telles que l'OSC.

[24]Il serait vain de vouloir citer l'ensemble de ces projets, surtout qu'une majeure partie semble rester fermée ou inaccessible. Mais un échantillon d'exemples est disponible sur le site du projet libpd.cc (site consulté en janvier 2018).

[25]Système d'exploitation mobile développé par Apple www.apple.com/fr/ios/ios-11 (site consulté en janvier 2018).

[26]L'application est disponible sur le site www.droidparty.net (site consulté en janvier 2018).

[27]Le plugiciel est originellement développé par l'utilisateur de Github "unknownError" et est publié depuis 2013 sur le répertoire Github github.com/unknownError/pdLV2-stereo (consulté en janvier 2018). Depuis 2016, il est mis à jour et maintenu par Alex Norman et accessible sur son répertoire Github github.com/x37v/pdlv2 (consulté en janvier 2018).

[28]Le plugiciel est développé par Karl Pannek and Oliver Greschke depuis juillet 2015 et est disponible sur le site pd-pulp.net (consulté en janvier 2018).

[29]Ce diagnostic est aussi fait lors des premiers essais de mise en œuvre de Camomile, cependant les développements et mises à jours parallèles de Pure Data permettront, comme cela sera présenté dans le suite de l'article, de proposer une solution honorable et intermédiaire avant d'arriver à une solution optimale avec les améliorations plus récentes de Pure Data.

[30]Le lien crca.ucsd.edu/~jsarlo/pdvst permettant d'accéder au projet initial n'est plus valide en 2018.

[31]Le plugiciel et les informations relatives au projet sont disponibles via le répertoire Github de Jean Yves Gratius github.com/jyg/pure-data (site consulté en janvier 2018).

Pure Data n'est pas réellement embarqué. Pour chaque instance du plugiciel, une instance de l'application de Pure Data est démarrée de telle sorte que chaque application possède un espace mémoire qui lui est propre, évitant dès lors des conflits[32]. Aussi, le plugiciel fonctionne en réalité comme un pont entre ces applications Pure Data associées aux plugiciels et la station de travail audionumérique. Le problème majeur reste que cet outil est restreint au système d'exploitation Windows. Il serait donc nécessaire d'étudier son potentiel fonctionnement sur d'autres systèmes d'exploitation, mais aussi de vérifier si cette approche n'implique pas de latence et si la communication entre les instances de Pure Data et la station de travail audionumérique via ce système est réellement stable, notamment dans un contexte de *multithreads*[33]. Une deuxième approche est proposée par la plateforme Heavy[34] distribuée par la société Enzien Audio, qui offre la possibilité de générer directement un plugiciel VST à partir d'un patch Pure Data. La plateforme offre un système de compilation en ligne, mais au lieu d'intégrer Pure Data au plugiciel, Heavy interprète le patch avec son propre moteur qui, quant à lui, offre un fonctionnement multi-instances. La force première de cette approche est aussi son principal défaut. En effet, la plateforme n'utilisant pas réellement Pure Data, elle ne propose qu'un sous ensemble de ses fonctionnalités, ce qui restreint les possibilités, et implique immanquablement un retard par rapport aux mises à jour de la distribution de Pure Data vanilla[35]. De plus, l'exacte similitude du rendu sonore entre le patch dans le logiciel Pure Data et le plugiciel généré n'est pas assurée, du fait de la réécriture totale ou partielle du moteur, afin de le rendre opérable dans un système multi-instances.

4. MISE EN ŒUVRE DE CAMOMILE

Lors de la définition des spécifications techniques et fonctionnelles de l'outil Camomile, deux enjeux majeurs sont apparus. Cette partie revient donc sur les choix opérés lors de mise en œuvre de l'outil Camomile d'une part, au niveau de l'intégration du moteur *patcher* dans le plugiciel et d'autre part, au niveau de la création d'un système de gestion réciproque entre le moteur *patcher* et le plugiciel. Cela en les confrontant à l'état de l'art présenté dans la partie précédente. À ce jour, deux versions majeures de Camomile ont été réalisées. Le changement de version correspond à une amélioration de la technologie utilisée. Sans pour autant revenir sur l'ensemble des spécificités de chaque version, les avantages et les conséquences de ces choix seront discutés.

4.1. Intégration du moteur dans le plugiciel

Pour les raisons évoquées précédemment, le plugiciel vise à utiliser le moteur de Pure Data qui offre une licence très libre[36] et un usage gratuit, tout en ayant une large communauté d'utilisateurs très actifs [12]. D'un autre côté, l'interface de programmation applicative JUCE[37] a été choisie, car elle offre entre autre la possibilité avec un seul code générique, de créer des plugiciels pour les formats VST et Audio Unit, ainsi que pour les systèmes d'exploitations Windows, Linux et MacOs. L'enjeu est alors de réussir à embarquer le moteur de Pure Data au sein du plugiciel. Or, comme cela a été évoqué précédemment, la difficulté première est de rendre le moteur de Pure Data compatible avec un usage multi-instances dans un contexte de *multithreads*.

4.1.1. L'approche séquentielle

Les développements de Pure Data pour la version 0.46, avec le début de l'intégration par Miller Puckette de la gestion du multi-instances, ont simplifié la première mise en œuvre de Camomile. Ces modifications du cœur du moteur, bien qu'insuffisantes à une mise en œuvre directe dans le contexte d'un plugiciel audio, ont permis de proposer une première stratégie consistant à forcer le caractère séquentiel des opérations. Pour cela, une interface similaire à libpd a été créée pour répondre spécifiquement à ce problème. La méthode consiste à restreindre l'accès au moteur à une instance à la fois, tout en limitant chaque instance à une seule opération, en utilisant un système de verrous (*locks*). Afin d'offrir un accès dans un contexte de *multithreads*, le système utilise des listes chaînées pour enregistrer les messages et les instructions qui sont lues et exécutées par la suite de manière séquentielle, avant de traiter le signal audionumérique de chaque instance. L'enjeu suivant est de simplement faire concorder l'interface JUCE avec l'interface enveloppant le moteur Pure Data. En avril 2016, une première version fonctionnelle destinée à un large usage[38] a pu être publiée. Elle a été testée sur de nombreuses plateformes logicielles et à chaque fois sur les systèmes d'exploitation pour lesquels elles sont disponibles[39]. Malgré l'approche radicale à forcer le

[32]Cette approche possède aussi l'avantage éventuel comparé aux approches présentées précédemment, de pouvoir utiliser directement l'interface graphique native. Il n'est néanmoins pas forcément souhaitable de vouloir utiliser l'interface graphique originale de Pure Data au sein d'un plugiciel audionumérique. Une interface spécifique et plus adaptée peut potentiellement être plus appropriée. Un autre avantage est la possibilité de charger les bibliothèques d'objets externes.

[33]Faute de moyens techniques et de temps, cela n'a pas pu être réalisé pour le moment.

[34]La plateforme Heavy est proposée par depuis 2016 sur enzienaudio.com (site consulté en janvier 2018).

[35]Distribution principale de l'application réalisée par Miller Puckette.

[36]Pure Data ou le projet libpd sont tous deux sous la même licence définie par "Standard Improved BSD" et distribuée avec les codes sources.

[37]JUCE est une interface de programmation applicative orientée vers le traitement du signal audionumérique distribué par la société Roli juce.com (site consulté en janvier 2018).

[38]La version 0.0.7 est toujours disponible sur le répertoire Github du projet.

[39]Une liste non exhaustive de ces plateformes – comprenant notamment les logiciels Reaper, Cubase, Ableton Live, Tracktion,

caractère séquentiel des opérations, le système était jouable et n'offrait pas d'artefacts en dehors d'une hausse de l'utilisation du processeur – résultat de l'attente de déverrouillage d'une instance par une autre.

4.1.2. L'approche parallèle

Plus récemment avec la version 0.47 de Pure Data, les derniers problèmes de l'utilisation multi-instances dans un contexte de *multithreads* ont été résolu en utilisant notamment la mémoire locale de *thread*. En gardant la mémoire associée à une instance locale sur chaque *thread*, cela permet d'utiliser en concurrence plusieurs instances sur des *threads* différents[40]. Néanmoins, l'usage d'une telle approche demande de revoir une grande partie de l'interface mise en œuvre précédemment. Aussi, afin que ce travail puisse être utile à une plus large communauté de programmeurs [14], le choix a été fait d'utiliser libpd et d'y intégrer ces améliorations. Les interfaces de programmation élémentaires de libpd pour le multi-instances ont été mises à jours notamment par Miller Puckette et Dan Wilcox. L'enjeu était alors de rendre certains aspects importants – notamment la réception des messages, des événements MIDI ou des notifications de console – compatibles avec le multi-instances. Aussi, la mise à jour pour le multi-instances et le *multithreads* du noyau de Pure Data n'a pas encore été réellement confrontée à un usage concret. Cette mise en œuvre était l'occasion de tester l'approche, et de corriger les quelques cas difficiles à prévoir[41]. En décembre 2017, la deuxième version de Camomile a pu être publiée et grâce au relatif succès de la précédente version, de nombreux retours d'utilisation constructifs ont permis de déployer rapidement les outils sur les différents systèmes d'exploitation[42].

4.2. Gestion réciproque entre le moteur et le plugiciel

La gestion du moteur *patcher* par le plugiciel et réciproquement des caractéristiques du plugiciel via le moteur, recouvre plusieurs aspects.

4.2.1. La communication et le partage de données

Un premier aspect important de la gestion du moteur par le plugiciel concerne la communication, et de manière générale le partage de données audio, MIDI ou encore de type message. La réception et l'envoi du signal audionumérique du plugiciel vers le moteur *patcher* et réciproquement, peuvent être réalisés respectivement avec les objets *adc~* et *dac~*. De manière analogue, les événements MIDI reçus par le plugiciel peuvent être récupérés dans un patch via les objets MIDI natifs de Pure Data. Et des événements MIDI peuvent être générés du patch et envoyés vers le plugiciel via les objets MIDI équivalents pour la sortie. Enfin, la réception de messages du plugiciel vers un patch est quant à elle réalisée avec les objets *receive* et les messages sont transmis du patch vers le plugiciel avec les objets *send*. Ces messages sont notamment utilisés pour modifier dans un patch les valeurs des paramètres mais aussi afin d'être notifié de leurs changements par la station de travail audionumérique (Figure 3)[43].

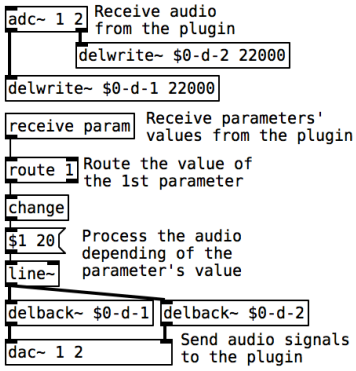

Figure 3. Patch Pure Data utilisé dans le plugiciel Bulgroz où le signal en entrée, reçu par l'objet *adc~*, est traité en fonction de la valeur du premier paramètre, reçu par l'objet *receive*, avant d'être renvoyé au plugiciel, via l'objet *dac~*.

L'avantage de cette approche est que, contrairement à Max For Live, aucun objet externe et spécifique à Camomile n'est nécessaire au fonctionnement d'un patch dans le plugiciel. Cela évite une possible obsolescence des patchs et les rend utilisables en dehors du plugiciel. Il est intéressant de remarquer qu'afin d'assurer une bonne distribution des messages entre les instances, la première version nécessitait l'utilisation des indices spécifiques aux patchs dans les symboles de réception et d'envoi[44]. La deuxième version avec sa gestion du multi-instances plus poussée, permet de se libérer de cette contrainte.

4.2.2. Le chargement des patchs

Un deuxième aspect notable de la gestion du moteur par le plugiciel, concerne la méthode de chargement des patchs. Deux approches peuvent être envisagées et ont été mises en œuvre respectivement dans la première et la deuxième version du plugiciel. La première version offrait la possibilité de charger des patchs dynamiquement via une boîte de dialogue. Cette approche possède l'avantage d'ouvrir rapidement un

Ardour, etc. – et des systèmes d'exploitation – Windows, MacOS et Linux en 32/64bits - est disponible sur le wiki du répertoire Github.

[40]L'accès à une même instance via plusieurs *threads* peut néanmoins toujours poser problème et l'accès sur un *thread* de plusieurs patchs est obligatoirement séquentielle.

[41]Les dernières modifications devraient être acceptées prochainement sur les branches principales de libpd et Pure Data.

[42]Cette version a déjà été testée sur un grand nombre de plateformes logicielles – telles que Reaper, Ableton Live, Tracktion, Bitwig, etc. – et de systèmes d'exploitation – Windows 32/64bits, MacOS 32/64bits et Linux 64bits.

[43]Ces approches peuvent être aussi utilisées pour la gestion des préréglages ou encore pour la configuration des canaux audio.

[44]Ces indices peuvent être intégrés en utilisant les caractères *$0* dans ces symboles.

patch dans la station de travail audionumérique. Associée avec la fonction de rechargement du patch courant, c'est une fonctionnalité très utile pour la création des patchs, mais qui pose néanmoins des problèmes. Plusieurs aspects du fonctionnement du plugiciel dépendent du patch chargé, comme les paramètres, les préréglages, les configurations de canaux supportées, *etc.* Or, changer de patch après le chargement initial du plugiciel par la station de travail audionumérique, implique de modifier ces éléments à la volée, ce qui n'est pas bien supporté par ces logiciels[45]. De plus, la localisation des patchs n'est pas assurée lorsque le projet de la station de travail audionumérique est partagé[46]. Enfin, utiliser l'interface graphique pour le chargement du patch empêche l'utilisation du plugiciel dans les stations de travail qui ne supportent justement pas les interfaces graphiques spécifiques aux plugiciels[47]. Pour répondre à ce problème, la deuxième version de Camomile a choisi d'associer chaque patch – ou un ensemble de patchs – à une nouvelle copie du plugiciel Camomile original. En liant et en intégrant de la sorte les patchs au fichier binaire du plugiciel, le chemin relatif entre les deux est toujours préservé. À la première instanciation du plugiciel dans la station de travail, le ou les patchs associés sont automatiquement chargés et l'ensemble des informations nécessaires au fonctionnement du plugiciel – paramètres, préréglages, configurations des canaux, *etc.* – peut être récupéré. Par la suite, le patch d'une instance du plugiciel peut être rechargé à la volée[48] – permettant donc de modifier le patch, que ce soit le moteur sonore ou l'interface graphique – mais les informations fournies à la station de travail audionumérique, et qui doivent nécessairement rester inchangées, ne sont pas modifiées.

4.2.3. Définition des propriétés du plugiciel

Enfin, un dernier aspect important concerne la manière de définir ces différentes informations liées au fonctionnement du plugiciel. Là encore, l'approche est différente entre les deux versions de Camomile. La première version se limitait aux seules informations des paramètres. Aussi, il avait semblé judicieux de définir les paramètres à l'aide des interfaces graphiques utilisateurs du patch principal. Par exemple, une glissière (*slider*) ou une boîte nombre générait automatiquement un paramètre en fonction du label et de la plage de valeur jouable définie dans les propriétés de l'objet graphique[49]. Cette approche possède l'avantage de savoir dans le plugiciel quand la valeur d'un paramètre est modifiée par l'utilisateur via cette interface, et de simplifier la mise en œuvre du plugiciel et la création du patch. Cependant, un tel choix possède des inconvénients majeurs. L'approche se complexifie dès lors que l'utilisateur souhaite utiliser plusieurs interfaces graphiques pour contrôler un même paramètre (Figure 2). Mais surtout, l'ordre des paramètres[50] dépend de l'ordre de création des interfaces graphiques. Aussi, dans le processus de création d'un patch pour le plugiciel, définir l'ordre des paramètres peut nécessiter de recréer les interfaces graphiques et leurs connexions. Ce qui peut être laborieux et source d'erreurs. Enfin il restait à définir de nouvelles méthodes pour configurer les autres options du plugiciel, telles que les préréglages, les configurations audio supportées, etc.

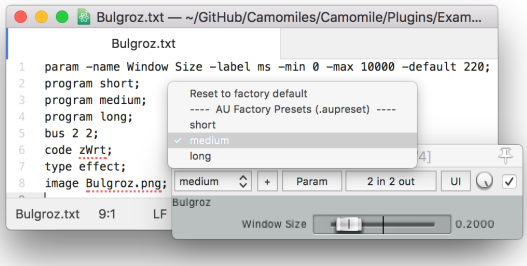

Figure 4. En arrière-plan, le fichier texte permettant de définir les propriétés du plugiciel Bulgroz avec Camomile, dont notamment le paramètre de taille de fenêtre et trois préréglages. Au premier-plan, l'interface graphique par défaut du plugiciel dans le logiciel Reaper permet de modifier la valeur du paramètre et de sélectionner les préréglages.

Afin de remédier à ces problèmes, la dernière version de Camomile utilise un fichier texte additionnel pour définir l'ensemble des informations nécessaires à un plugiciel. La syntaxe de ce fichier est simple. Chaque ligne permet de définir une option, de rajouter un paramètre ou un préréglage[51] ce qui ouvre la possibilité d'écrire ou de charger le fichier via l'objet *text* de Pure Data (Figure 4. En arrière-plan, le fichier texte permettant de définir les propriétés du plugiciel Bulgroz avec Camomile, dont notamment le paramètre de taille de fenêtre et trois préréglages. Au premier-plan, l'interface graphique par défaut du plugiciel dans le logiciel Reaper permet de modifier la valeur du paramètre et de sélectionner les préréglages.). Enfin, un système de

[45]Cette approche est normalement interdite par la norme VST. Une restriction est définie dans la documentation du VST 3 Plug-In SDK github.com/steinbergmedia/vst3sdk (page consultée en janvier 2018).

[46]Les patchs n'étant pas directement associés au projet en cours, au fichier binaire du plugiciel ou à la station de travail, son chemin est absolu et lorsqu'un projet est partagé sur un autre ordinateur, le plugiciel ne parvient pas à retrouver le patch et il est nécessaire de redéfinir manuellement tous les chemins.

[47]Certaines stations de travail audionumériques supportent encore mal les interfaces graphiques des plugiciels. C'est le cas notamment du logiciel libre et gratuit Audacity audacity.fr (page consultée en janvier 2018).

[48]Cette fonctionnalité peut être très utile notamment lors de la création du plugiciel. La version 1.0.4 du plugiciel offre, pour cela, la possibilité de recharger le patch manuellement – en cliquant sur un bouton dédié – ou automatiquement à chaque nouvelle sauvegarde du patch.

[49]Cette approche est aussi définie dans la documentation de la version 0.0.7 du plugiciel disponible sur le wiki du répertoire Github.

[50]L'ordre des paramètres est très important, car les stations de travail audionumérique utilisent leurs indices (ou positions) – et non leurs noms – comme identifiants des paramètres pour organiser les informations qui leurs sont relatives, telles que les automations.

[51]Cette approche est définie dans la documentation du plugiciel et sera développée plus longuement dans une prochaine publication.

notifications du plugiciel vers le patch et réciproquement du patch vers le plugiciel, permet d'associer les interfaces graphiques aux paramètres[52]. La version actuelle du plugiciel permet donc un support complet de l'ensemble des fonctionnalités génériques et des spécifications des plugiciels, tout en préservant une cohérence avec le langage et les usages de Pure Data.

5. BILAN ET PERSPECTIVES

Camomile offre à présent un outil compatible sur les trois principaux systèmes d'exploitation avec la plupart des stations de travail audionumérique supportant des plugiciels. Les choix opérés dans la dernière version du plugiciel – que ce soit pour le chargement de patchs, la communication entre le moteur et le plugiciel ou encore la création de patchs ainsi que la définition des propriétés du plugiciel – ont permis de supporter pleinement l'ensemble des fonctionnalités usuellement offert par ce type de formats : la gestion des paramètres, des préréglages, des configurations audio, des événements MIDI, des informations de la tête de lecture[53] ou encore la création d'une interface graphique adaptée et fonctionnelle. Ainsi, des patchs correctement configurés et tirant parti de l'ensemble de ces fonctionnalités, peuvent permettre de générer des plugiciels rivalisant que ce soit sur l'aspect sonore ou l'aspect ergonomique avec des plugiciels codés et compilés de manière classique. De plus, l'outil n'est pas seulement limité à un usage expérimental, mais peut très bien être utilisé dans un contexte de production musicale abouti ou dans un contexte pédagogique[54] (Figure 5).

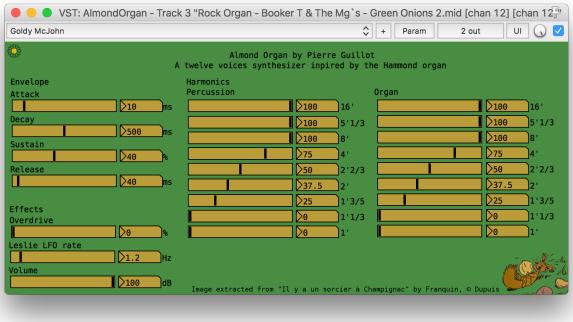

Figure 5. Le plugiciel *AlmondOrgan* au format VST, un synthétiseur douze voix inspiré par l'orgue Hammond, créé avec Camomile et intégrant de multiples fonctionnalités comme la gestion des événements MIDI, des paramètres et des préréglages.

Camomile est, en outre, un outil extrêmement intéressant pour les développeurs car il permet de créer rapidement des prototypes de plugiciels en prenant en compte tous les aspects de la mise en œuvre : moteur audio, interfaces graphiques, messages, *etc.* Et au-delà des prototypes, les plugiciels créés avec Camomile peuvent être publiés en tant que tels. Le plugiciel jouit déjà d'une relative notoriété dans la communauté d'utilisateurs de Pure Data et de l'audionumérique libre de manière générale. Mais il serait intéressant d'ouvrir la communauté d'utilisateurs via le partage direct de plugiciels créés avec Camomile. Cela pourrait donner envie à des utilisateurs exclusifs des stations de travail audionumérique de s'intéresser aux approches de type *patcher*. Sur le plan du développement, Camomile aura été un excellent terrain d'expérimentations et de mise à l'épreuve du support *multithreads* et multi-instances de Pure Data. Il aura permis de révéler un certain nombre de problèmes qui restaient à corriger. Enfin, certains points sont néanmoins encore à explorer et d'autres problèmes à débloquer. C'est le cas notamment de la gestion des bibliothèques externes d'objets qui nécessitent *a priori* d'être compilées avec le plugiciel. Il existe deux raisons à cela. D'une part, les stations de travail audionumériques semblent pour la plupart bloquer le chargement dynamique de bibliothèques externes. Et d'autre part, il est nécessaire de vérifier si les bibliothèques externes n'interfèrent pas avec les conditions d'activation du support du multi-instances et du *multithreads* de Pure Data. Il serait aussi intéressant de réfléchir à une amélioration ou une extension de la création et de la gestion de l'interface graphique. Il s'agirait de prendre en compte le système de structure de données graphiques [16] ou d'offrir un système dynamique permettant d'intégrer ses propres graphismes aux interfaces, afin de se rapprocher d'un plugiciel plus classique[55]. Enfin, il serait intéressant d'intégrer la génération de plugiciels au format LV2[56] mais aussi au format Audio Unit v3[57] qui permettrait de charger les plugiciels sur des tablettes et téléphones iOS.

6. REMERCIEMENTS

Je tiens à remercier toute la communauté de développeurs de Pure Data et de libpd, et particulièrement Miller Puckette et Dan Wilcox, pour leurs conseils et leurs explications mais aussi les

[52]Cette approche est aussi définie dans la documentation du plugiciel et sera aussi développée plus longuement dans une prochaine publication.

[53]Il est en effet possible d'utiliser toutes les informations fournies par la station de travail audionumérique telles que le chiffrage de la mesure, le tempo de la mesure, la position de la tête de lecture, etc.

[54]Le plugiciel a notamment été présenté à l'Université Paris 8 aux étudiants en 2016 et 2017, dans le cadre des cours d'Introduction à la programmation avec Max et Pure Data 1 dispensés par moi-même et en 2018, aux étudiants du cours de Composition électroacoustique 2 dispensé par Alain Bonardi et suite auquel ils ont été invité à créer leurs propres plugins.

[55]Le plugiciel offre déjà à la possibilité de charger des images de fond (Figure 2 & Figure 5) mais il serait intéressant de pouvoir replacer les graphismes natifs de objets de Pure Data, tels que l'interrupteur (*toggle*), la glissière (*slider*) ou encore la boîte nombre (*numbox*), en utilisant des séries d'images données par l'utilisateur.

[56]Le choix a été fait pour le moment d'attendre que JUCE supporte ce format plutôt que coder un plugiciel indépendant spécifique.

[57]Grâce au support de ce format par JUCE, il peut être déjà envisagé de publier une version Audio Unit v3 de Camomile mais faute de temps et de moyen techniques il n'a pas encore été possible de tester l'approche.

utilisateurs de Camomile pour leurs retours d'utilisation et leurs suggestions. Je tiens aussi à remercier toute l'équipe du CICM pour l'intérêt porté à ce projet, et particulièrement Alain Bonardi et Eliott Paris pour leurs remarques et leurs suggestions.

7. REFERENCES

[1] Puckette M. "Pure Data : Another Integrated Computer Music Environment", Proceedings of the Second Intercollege Computer Music Concerts, p. 37-41, Tachikawa, Japon, 1997.

[2] De la Hogue T., Baltazar P., Catherine M. D., Chao J. et Bossut, C. "Ossia : Open Scenario System For Interactive Applications", actes des Journées d'Informatique Musicale, p.78-84, Bourges, France, mai 2014.

[3] Coduys T., Lefèvre A. et Pape G. "IanniX", actes des Journées d'Informatique Musicale, Montbéliard, France, juin 2003.

[4] Sèdes A., Guillot P. et Paris E. "The HOA library, review and prospect", Proceedings of the ICMC-SMC 2014, p. 855-860, Athènes, Grèce, septembre 2014.

[5] Guillot P., Paris E. et Deneu M. "La bibliothèque de spatialisation HOA pour Max/MSP, Pure Data, VST, FAUST…", Revue Francophone d'Informatique et Musique (En ligne), n° 3, automne 2013, URL : revues.mshparisnord.org/rfim/index.php?id=245.

[6] Favreau E., Fingerhut M., Koechlin O., Potacsek P., Puckette M. et Rowe R., "Software Developments for the 4X Real-time System", Proceedings of the International Computer Music Conference 1986, p. 369-373, La Haye, Pays-Bas, 1986

[7] Guillot P. "La Représentation Intermédiaire et Abstraite de l'espace Comme Outil de Spatialisation du Son", thèse de doctorat, CICM, Université Paris 8, Saint-Denis, France, 2017.

[8] Paris E., Millot J., Guillot P., Bonardi A. et Sèdes A. "Kiwi : Vers un Environnement de Création Musicale Temps-Réel Collaboratif - Premiers Livrables du Projet Musicoll", Actes des Journées d'Informatique Musicale, Paris, France, mai 2017.

[9] Puckette M. "The Patcher", Proceedings of the International Computer Music Conference 1988, p. 420-429, Cologne, Allemagne, 1988.

[10] Puckette M. "Combining Event and Signal Processing in the MAX Graphical Programming Environment", Computer Music Journal, vol. 15, n° 3, p. 68-77, automne 1991.

[11] Zicarelli D. "Developing Plug-ins in Max/MSP for Pluggo - updated information for Pluggo 3", Documentation du plugiciel Pluggo publiée par Cycling'74, révision 4 du 16 mai 2002.

[12] Puckette M. "Who owns our software ? A First-person Case Study", Proceedings of the International Symposium on Electronic Art, p. 200-202, Helsinki, Finlande, 2004.

[13] Geiger G. "PDa : Real Time Signal Processing and Sound Generation on Handheld Devices," Proceedings of the International Computer Music Conference, Singapour, septembre-octobre 2003.

[14] Brinkmann P., Kirn P., Lawler R., McCormick C., Roth M. et Steiner H.-C. "Embedding Pure Data with libpd", Proceedings of the Pure Data Convention, Weimar, Allemagne, aout 2011.

[15] Wilcox D., "PdParty : An iOS Computer Music Platform using libpd", Proceedings of the Pure Data Convention, New-York, États-Unis, novembre 2016.

[16] Puckette M. "Using Pd as a score language", Proceedings of the International Computer Music Conference (ICMC 2002), p. 184-187, Göteborg, Suède, 2002.