

KIWI : VERS UN ENVIRONNEMENT DE CREATION MUSICALE TEMPS REEL COLLABORATIF PREMIERS LIVRABLES DU PROJET MUSICOLL

Eliott Paris, Jean Millot, Pierre Guillot, Alain Bonardi, Anne Sèdes

▶ **To cite this version:**

Eliott Paris, Jean Millot, Pierre Guillot, Alain Bonardi, Anne Sèdes. KIWI : VERS UN ENVIRONNEMENT DE CREATION MUSICALE TEMPS REEL COLLABORATIF PREMIERS LIVRABLES DU PROJET MUSICOLL. Journées d'Informatique Musicale 2017, May 2017, Paris, France. hal-01550190

HAL Id: hal-01550190
https://hal.science/hal-01550190

Submitted on 29 Jun 2017

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.

KIWI : VERS UN ENVIRONNEMENT DE CREATION MUSICALE TEMPS REEL COLLABORATIF

PREMIERS LIVRABLES DU PROJET MUSICOLL

Eliott Paris
CICM - EA1572, Université Paris 8, Labex Arts-H2H
eliottparis@gmail.com

Jean Millot
CICM - EA1572, Université Paris 8
jean.millot7@gmail.com

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

Alain Bonardi
CICM - EA1572, Université Paris 8 et Ircam
alain.bonardi@univ-paris8.fr

Anne Sèdes
CICM - EA1572, Université Paris 8
anne.sedes@univ-paris8.fr

RÉSUMÉ

Cet article présente l'ensemble du projet MUSICOLL (MUSIque temps réel,[1] COLLaborative et nomade), qui vise à repenser la création audionumérique en musique temps réel dans sa dimension collaborative. Nous décrivons l'état actuel de l'application Kiwi[2], logiciel issu des travaux de recherche et développement de MUSICOLL. Nous montrons ensuite les choix faits en termes de conception collaborative, fonctionnelle et technique, ainsi que les différentes perspectives de recherche.

1. INTRODUCTION

Des années 1950 à nos jours, les configurations socio-technologiques de travail des compositeurs pratiquant la musique électronique ont bien évolué [9]. La période actuelle est marquée par la triple évolution apportée par le nomade, le cloud et le collaboratif :

- d'une part le développement des tablettes et périphériques nomades qui proposent une autre relation à l'informatique et renouvellent l'interaction gestuelle avec le musical (*multitouch*, accéléromètres…) ;
- d'autre part, la diffusion sur Internet d'applications créatives dans le domaine du son, offrant au grand public des fonctionnalités auparavant réservées à des experts. Par exemple, les applications de montage sonore foisonnent : on en recense plus de 280 pour iPad sous l'appellation « séquenceur » ;
- enfin, une évolution techno-sociologique marquée par la mise en réseau des créateurs, des produits de leur création[3] et de leurs outils, qui deviennent collaboratifs, sur la base du cloud computing. Après le partage des productions (photos, films, morceaux musicaux) vient le partage des savoir-faire. Par exemple, le groupe réseau *PowerBooks Unplugged* a développé le système *Republic* en SuperCollider pour permettre d'écrire simultanément et de manière collaborative du code distribué [4].

Tirant parti de cette diffusion et de cette re-configuration technologique et sociétale, de nouvelles pratiques et de nouveaux outils collaboratifs sont proposés, de la performance à la création, en passant par la lutherie :

- performance collective avec des instruments prédéfinis sur des supports numériques connectés comme dans les projets SmartFaust[4] ou de Cosima[5] [1];
- partage et pilotage collectif d'un même moteur sonore, non éditable collaborativement, comme dans le cas de Miraweb[6] ;

[1] Le terme temps réel est utilisé ici et dans la suite du texte pour décrire les pratiques musicales impliquant la création d'un dispositif capable de traiter le son en temps réel comme dans le cas de logiciel de *patching* audio de type Max ou PureData.

[2] https://github.com/Musicoll/Kiwi/releases/latest

[3] Via les environnements sociaux de diffusion de contenus musicaux de type https://soundcloud.com/

[4] http://www.grame.fr/prod/smartfaust

[5] http://cosima.ircam.fr/

[6] https://github.com/Cycling74/miraweb

- édition collaborative de contenus avec un séquenceur audionumérique comme le logiciel Ohm Studio[7], conçu pour faire du montage audio à plusieurs en réseau ;
- Au sein du McGill Digital Orchestra, un *framework* collaboratif a été conçu pour permettre le co-développement d'instruments de musique numériques associant contrôleurs et synthétiseurs [10].

Dans le cas des environnements de traitement temps réel du son, l'essentiel des pratiques collectives s'appuie sur des espaces de stockage communs sur le cloud permettant d'échanger des productions. Il n'y a en général pas de partage de l'espace de travail, ce qui est pourtant fondamental dans les usages de création collective [2]. Pourtant, l'aspiration à la création collaborative est bien présente, entraînée par l'envie d'apprendre : parmi les retours qu'ils obtiennent des utilisateurs, les concepteurs du logiciel Ohm Studio retrouvent souvent l'idée que le choix de rejoindre un projet collectif en ligne est motivé par l'ouverture à une autre forme d'apprentissage.

Le point de départ de notre démarche est la situation pédagogique de notre cours d'introduction à la programmation graphique en audionumérique, sur Max et Pure Data, dans le cadre de la mineure « composition assistée par ordinateur », de la licence « Musicologie et pratique musicale » du département de musique de l'Université de Paris 8. Actuellement, l'enseignant connecte son ordinateur à un vidéoprojecteur et à un système-son pour un cours collectif, commençant par des notions élémentaires : qu'est-ce qu'un oscillateur, un gain, une sortie audio ?

Ce dispositif mérite d'être renouvelé. En effet, s'il a bien fonctionné depuis la fin des années 1990, l'accès à Internet, aux forums et aux réseaux sociaux est désormais omniprésent en classe et il n'est pas rare de voir les étudiants vérifier la parole et la démonstration de l'enseignant, par exemple sur Wikipédia ou des tutoriels YouTube, plutôt que de poser une question oralement. Face à ces usages qui mettent en concurrence le savoir de l'enseignant avec les données du Web, il nous semble intéressant de repenser la pédagogie en proposant une dynamique d'apprentissage collectif fondée sur une approche collaborative en réseau [11].

En conséquence, nous développons dans le cadre du projet MUSICOLL une recherche sur la programmation graphique collaborative en audionumérique visant à re-penser sa pratique, son utilisation artistique, et son enseignement. L'ensemble du projet est exposé dans la première partie de l'article. Nous détaillons ensuite l'application Kiwi, issue des recherches menées dans le projet MUSICOLL, en commençant par ses fonctionnalités collaboratives montrées dans un cas d'utilisation simple, puis en décrivant le détail de son implémentation, en justifiant nos choix fonctionnels et techniques. Dans un dernier temps, nous présentons nos perspectives de recherche.

2. PRÉSENTATION GLOBALE DU PROJET MUSICOLL

L'idée d'un projet de recherche mené par le CICM[8] sur ces thématiques est née du constat de l'émergence des pratiques collaboratives et du faible nombre de réalisations permettant leur mise en place dans le contexte du *patching* audio qui est le nôtre. Il est né aussi des observations que nous avons effectuées sur les environnements existants de traitement temps réel du son en menant nos projets de recherche précédents autour de la spatialisation ambisonique [3] ou encore de notre pratique de l'enseignement de Max[9] et Pure Data[10] à l'Université Paris 8. Pour mener à bien ce projet, une solution envisageable aurait été de partir de l'un de ces logiciels et d'y *ajouter* une dimension collaborative ; cette solution n'a pas été retenue dans la mesure où dans le cas de Pure Data, un refactoring isofonctionnel fastidieux aurait été nécessaire, et dans le cadre de Max, bien que disposant d'une API ouverte, celle-ci reste limitée et nous aurait freinés dans le développement [7][11]. Nous avons donc opté pour la conception de notre propre logiciel de *patching* audio.

Le projet ANR MUSICOLL[12] (2016-2018) associe le CICM et la société Ohm Force, dont l'une des spécialités est l'audio collaboratif, dans une recherche consacrée à la musique temps réel collaborative et nomade[13], avec les objectifs suivants :

- la spécification et la production de l'environnement Kiwi de traitement temps réel collaboratif et nomade, organisées en cycles courts ;
- l'étude de la prise en main de cette plateforme par des créateurs ;
- l'étude du renouvellement induit dans l'enseignement des logiciels de traitement temps réel du son, en proposant la refonte du cours de Max et Pure Data pour débutants de Licence 2 au Département Musique de l'Université Paris 8 ;

[7] https://www.ohmstudio.com/

[8] Centre de Recherche en Informatique et Création Musicale, Laboratoire Musidanse (EA 1572), Université Paris 8.

[9] https://cycling74.com/products/max/

[10] http://puredata.info/

[11] Comme présenté dans la quatrième partie de cet article, la mise en place de l'aspect collaboratif dans un logiciel nécessite d'être prise en compte au cœur même du modèle de données traité par celui-ci. Cela amène impérativement des problèmes de compatibilité si le logiciel est dérivé d'un logiciel mère, comme Pure Data par exemple, et fait perdre ainsi la majorité des atouts apportés par ce type d'approche, temps de développement, communauté d'utilisateurs préexistante, etc.

[12] En réponse à l'appel à projets générique 2015 de l'ANR.

[13] Par nomade, nous entendons aussi bien la machine-support en ouvrant à des appareils qui ne sont pas des postes fixes, par exemple des tablettes ou des systèmes embarqués tels que (*raspberry*, *bela*, etc). ou encore l'utilisation du logiciel dans une version plugin au sein d'un logiciel hôte.

- la dissémination scientifique et auprès des professionnels.

Quatre axes de recherche et développement guident les travaux menés sur le projet :

- la prise en compte de **l'apprentissage** comme dimension fondamentale du collaboratif [5], que ce soit au niveau des créateurs ou au niveau de l'enseignement des logiciels de traitement temps réel du son.
- les **représentations de l'espace de travail collaboratif** et la navigation : nous réfléchissons à des représentations à la fois proches de celles existantes dans Max et Pure Data, mais aussi à des représentations hiérarchiques des objets et connexions créés, organisés en arborescence selon les patchs et sous-patchs. Les questions d'association dans le même espace de travail ou au contraire de séparation du traitement et du contrôle sont également posées.
- les **représentations du temps dans les processus de travail collaboratif** : notre réflexion porte sur la projection temporelle et la visualisation des structures hiérarchiques modélisant le travail collaboratif pour saisir le processus de création partagé.
- la **pérennité des résultats du travail collaboratif** : grâce à l'accès et au stockage des patchs en ligne, à la mise à disposition d'un historique des modifications ou encore à un premier système d'annotation qui est actuellement à l'étude et qui devrait permettre d'enrichir la description du traitement élaboré à plusieurs mains.

Après environ un an de travail sur le projet, nous disposons d'une première maquette du logiciel Kiwi développée en langage C++, associée à un premier ensemble de spécifications et d'un scénario simple de cours sur la synthèse additive.

3. PRÉSENTATION DE L'ÉTAT ACTUEL DE L'APPLICATION KIWI

Kiwi est un outil de *patching* audio graphique dans la lignée de logiciels tels que Pure Data ou Max, auquel vient s'ajouter une dimension collaborative permettant aux utilisateurs de travailler à plusieurs en réseau sur un même traitement ou une même composition audionumérique. Un des objectifs de la phase initiale du projet MUSICOLL était la création d'une première maquette de Kiwi. Nous présentons ici l'interface de cette maquette et son utilisation dans un contexte collaboratif. Kiwi étant destiné à être utilisé dans un premier temps dans un contexte d'enseignement universitaire, cette première maquette a été élaborée dans le but de répondre à un scénario de cours de débutant en programmation audionumérique, consacré à la synthèse additive.

La première version de Kiwi permet d'ores et déjà une utilisation collaborative bien que restreinte sur certains aspects afin de faciliter les éventuelles refontes nécessaires dans une phase de prototypage : le nombre d'objets a volontairement été limité et son utilisation à plusieurs n'est pour le moment permise qu'au sein d'un réseau local.

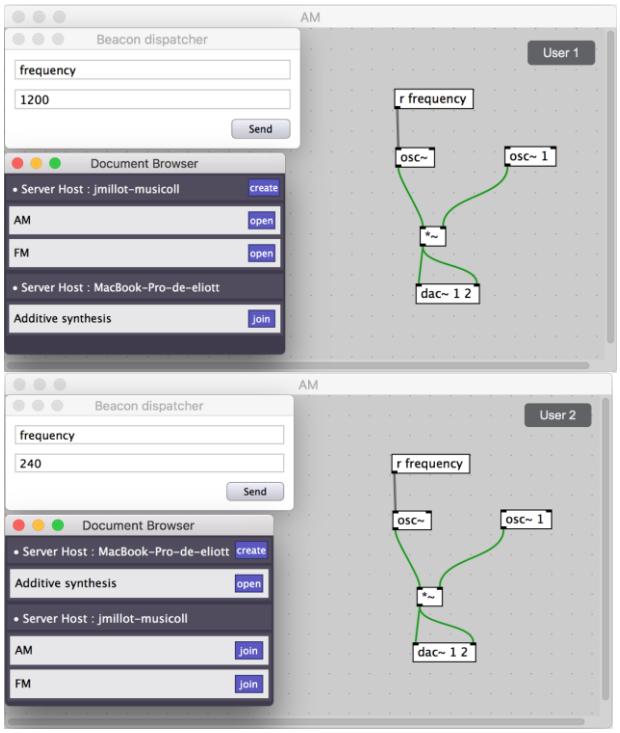


Figure 1. Vues de l'espace de travail chez deux utilisateurs qui éditent ensemble le même patch « AM » au sein de l'application Kiwi et gèrent le contrôle de manière différencié.

L'interface de Kiwi (figure 1) permet de visualiser et modifier l'ensemble des documents partagés par les utilisateurs sur le réseau (fenêtre Document Browser). Le contrôle du patch, qui reste pour l'instant local, est quant à lui externalisé dans une fenêtre (*Beacon Dispatcher*) associant un nom présent dans le patch (*frequency*) et une valeur (1 200 pour l'utilisateur 1 et 240 pour l'utilisateur 2)

4. CONCEPTION COLLABORATIVE DE L'APPLICATION KIWI

Sur le plan collaboratif, la première difficulté réside au niveau technique : comment faire en sorte que plusieurs utilisateurs puissent travailler sur le même patch au même moment sans créer de conflits ? L'architecture réseau de Kiwi est de type client-serveur. Le serveur, unique pour chaque patch édité, centralise les connexions clients, dispose du patch et se charge de maintenir sa cohérence au cours de son édition collaborative. Un ou plusieurs clients peuvent se connecter au patch sur un serveur et disposer chacun d'une version locale temporaire de celui-ci qu'ils peuvent éditer directement tout en restant synchronisés avec la version distante. Pour maintenir synchronisées entre elles chacune des versions locales du

patch, ces deux applications utilisent un même modèle de données que nous avons conçu à l'aide de Flip, *framework* propriétaire développé par la société Ohm Force[14]. Flip permet de maintenir la cohérence d'un modèle de données, appelé document ou patch dans le cas de Kiwi, partagé et édité de manière concurrente sur un réseau. Pour cela, chaque modification apportée par un client à son document local (qui se manifeste pour nous par l'ajout d'un objet, son déplacement au sein du patch, l'ajout ou la suppression d'un lien…) est appliquée puis traduite en une transaction qui est alors envoyée au serveur pour être validée. Si cette transaction est jugée invalide par le serveur, celle-ci est rejetée puis annulée côté client ; si elle est valide en revanche, celle-ci est appliquée au document sur le serveur puis transmise aux autres utilisateurs connectés afin que leur version locale du document se synchronise avec la version distante en appliquant à leur tour la transaction.

Lorsque l'on édite un patch à plusieurs, des actions peuvent être réalisées simultanément, provoquant potentiellement des conflits. Dans les outils collaboratifs de type *git*[15] ou *Dropbox*[16], les conflits provoquent le plus souvent une perte d'information ou au mieux, leur résolution est confiée aux utilisateurs eux-mêmes : il leur revient alors d'indiquer manuellement quelle version est valide ou d'adapter le document pour qu'il le devienne. L'avantage de l'API Flip est qu'elle propose un mécanisme de résolution automatique des conflits. Cette dernière se base sur un procédé de validation permettant de détecter les incohérences générées par l'exécution de transactions conflictuelles et de garantir ainsi une validité permanente du document. Dans quel état doit par exemple se trouver le patch quand un utilisateur choisit de supprimer un objet pendant qu'un autre tente de manière concurrente de le connecter en créant un lien à partir de celui-ci ? Si ces deux transactions étaient appliquées sans validation, un lien ne pointant vers aucun objet pourrait être créé, provoquant alors une incohérence au sein du document. Dans notre cas, le serveur est capable de détecter ce genre d'anomalie et de s'y adapter en rejetant les transactions qu'il juge incorrectes. Le résultat dépendra donc de l'ordre de réception (la première modification étant celle retenue), menant soit à la création du lien, soit à la suppression de l'objet.

Au-delà de la modification concurrente du modèle de données, d'autres enjeux sont directement liés au développement d'un environnement logiciel collaboratif. L'interface graphique doit par exemple être repensée afin de permettre à chaque utilisateur d'être informé des actions passées ou présentes des autres utilisateurs sur un patch sans pour autant diminuer la lisibilité générale de celui-ci. Le prototypage de l'application fera appel dans les prochaines étapes aux concepts issus des travaux de la communauté IHM, notamment sur les interfaces collaboratives [8].

La sélection des objets et des liens au sein du patch est un exemple d'information que nous avons choisi de partager entre les utilisateurs. Ce problème peut être géré différemment selon les cas. Par exemple, dans la suite d'application en ligne Google Docs[17], chaque utilisateur est représenté par une couleur de curseur et de sélection différente. Cette solution offre l'avantage de pouvoir distinguer et situer clairement chaque utilisateur au sein du document, mais a tendance à « polluer » l'espace de travail si trop de personnes y sont connectées.

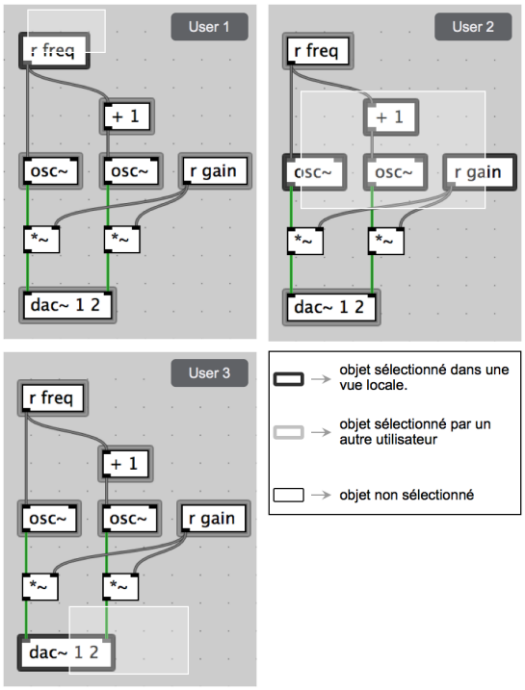


Figure 2. Représentation (noir et blanc) de la sélection de trois utilisateurs interagissant simultanément au sein du même patch Kiwi.

Nous avons pour l'instant considéré que savoir si quelqu'un sélectionne actuellement un objet particulier est plus important que de savoir qui a effectivement sélectionné cet objet. Le choix qui a été arrêté aujourd'hui, inspiré par l'approche déjà en place dans le logiciel Ohm Studio pour la sélection des *clips*, permet de distinguer simplement les objets sélectionnés localement des objets sélectionnés par d'autres personnes sur le réseau grâce à une couleur de bordure différente pour les deux cas (figure 2).

La sélection des objets n'est pas le seul problème d'ergonomie collaborative auquel nous sommes confrontés. L'interface du logiciel doit être capable de rendre compte le plus distinctement possible des actions

[14] http://irisate.com/flip-overview/ (Irisate est une compagnie fondée par plusieurs associés de la société OhmForce).

[15] https://git-scm.com/

[16] https://www.dropbox.com/

[17] https://www.google.com/drive/

réalisées par les autres collaborateurs. Dans les premières versions de Kiwi, le fait qu'un utilisateur supprime un objet ou le déplace en dehors de la zone visible du patch produisait par exemple exactement le même résultat visuel chez les autres utilisateurs connectés : ils le voyaient simplement disparaître sans pouvoir identifier clairement l'action qui s'était déroulée. Pour remédier à ce problème, nous différencions maintenant ces actions en animant le déplacement de l'objet et en appliquant une animation de type fondu à la suppression.

Actuellement les sélections et les noms des documents joignables sur le réseau local sont les seules données partagées entre les utilisateurs en dehors des objets et des liens. Les prochains développements fourniront davantage d'informations, qui devront par exemple permettre de savoir qui a rejoint ou quitté un document, qui a édité tel ou tel objet ou réalisé telle ou telle action. Pour permettre l'identification de ces actions nous pourrons passer par exemple par la création d'une vue dédiée à un historique des modifications réalisées dans le temps par les différents collaborateurs au sein du patch. Cet historique pourrait aussi permettre aux utilisateurs de charger une version antérieure du patch ou encore servir à visualiser l'évolution d'un patch au cours de son élaboration dans un contexte pédagogique. Plus largement se posent les questions d'attribution de rôles aux différents participants de la création collaborative, distinguant le créateur du document de celui qui l'a rejoint ensuite, pour aboutir à une première approche des espaces privé et public au sein de l'application, concepts importants dans ce type de logiciel [6].

L'intégration de solutions de communication est aussi à l'étude comme la mise en place d'un système de commentaires de type *post-it*. Les utilisateurs pourraient ainsi les disposer sur le patch pour ajouter une note, décrire le processus en cours, ou encore demander de l'aide ou une révision à une autre personne par l'intermédiaire de notifications.

L'application n'a pour le moment été testée qu'au sein d'un réseau local. Dans ce mode chaque client embarque avec lui son propre serveur auquel peuvent se connecter les autres utilisateurs du réseau. Une prochaine étape de développement ouvrira la possibilité pour les clients de se connecter à un serveur distant par Internet. Cela nécessitera l'authentification des utilisateurs ainsi que le stockage et l'accès aux patchs et ressources à distance qui est plus difficile à mettre en place dans une architecture réseau décentralisée [13]. Nous serons alors amenés à gérer le cas de synchronisation de documents édités en mode hors-ligne par les utilisateurs.

Une autre question discutée actuellement au sein du projet est celle du partage des valeurs de jeu du patch. Actuellement seules les données d'édition (type d'objet ajouté, place des objets au sein du patch, connexions entre les objets…) sont partagées entre les utilisateurs. Or il pourrait être intéressant de récupérer les valeurs de jeu d'un utilisateur donné pour entendre ce qu'il entend, tester d'autres réglages du patch à l'aide d'un système de preset partagé ou encore contrôler un patch à distance comme le permettent déjà des solutions actuelles telles que MiraWeb pour Max.

5. CHOIX FONCTIONNELS ET TECHNIQUES GÉNÉRAUX

Ce projet nous donne aussi l'occasion de questionner certaines spécifications plus générales relatives aux logiciels de *patching* audio : la mise en œuvre de la chaîne DSP, la représentation des objets graphiques, le fonctionnement des arguments des objets ou encore la gestion des erreurs.

Le modèle de la chaîne DSP a été élaboré en accordant une importance particulière aux performances tout en offrant la possibilité de fonctionner en parallèle, dans un système multi-thread, permettant ainsi la portabilité vers d'autres plateformes et le fonctionnement au sein de plug-in audio (VST, AudioUnit, etc.).

À terme, le logiciel proposera un ensemble d'objets graphiques[18], tels que les *bang*, *toggle*, *slider*, etc. disponibles dans les logiciels Max et Pure Data, offrant l'avantage d'être déjà bien établis tant sur le plan des spécifications de développement que d'un point de vue usage. Cependant, pour l'instant leur mise en œuvre a été repoussée et les objets temporairement remplacés par une fenêtre permettant d'envoyer des messages au patch.

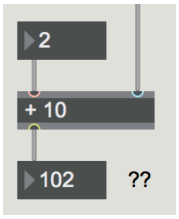


Figure 3. Représentation d'un patch Max dans lequel l'argument initial peut produire une confusion avec le résultat du calcul.

Sur un plan pédagogique, il nous semble nécessaire de revoir la gestion et la représentation des arguments des objets proposées dans les logiciels Max et Pure Data afin que ce qui est visible à l'utilisateur représente au mieux le fonctionnement interne du patch (figure 3). Ainsi, le choix a été fait d'interpréter un argument comme une constante de l'objet si celui-ci est spécifié à sa création.

[18] Un objet graphique est un objet offrant une interface de visualisation et/ou d'interaction particulière à l'utilisateur.

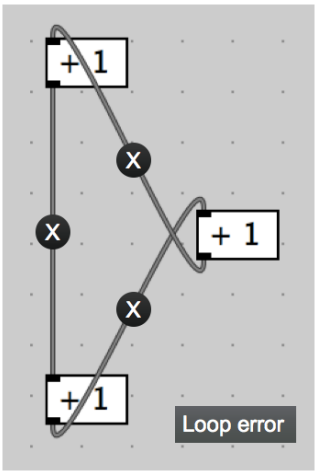


Figure 4. Visualisation souhaitée d'une erreur de type débordement de pile au sein d'un patch Kiwi.

La signalisation des erreurs de conception du patch par l'utilisateur est également un point sur lequel nous portons une attention particulière. Nous souhaitons par exemple pouvoir remonter le flux des objets et des liens lors d'un débordement de pile (figure 4) ou lors de la création d'une boucle dans le DSP. Bien que n'étant pas encore disponible à l'utilisateur final dans la version actuelle, le mécanisme est déjà pensé au niveau du code.

6. PERSPECTIVES DE RECHERCHE

Comme nous l'avons montré, les premières versions du logiciel sont d'abord destinées à deux usages privilégiés : la pédagogie pour les débutants d'une part et la prise en main par les créateurs d'autre part.

6.1. Kiwi dans un contexte de pédagogie du temps réel

Comme évoqué dans l'introduction de cet article, nous souhaitons reconfigurer le cours d'introduction à la programmation graphique audionumérique à partir des recherches menées dans le cadre du projet MUSICOLL.

Ce cours a été donné à tour de rôle par plusieurs des auteurs de cet article qui ont accumulé une bonne expérience pédagogique de l'enseignement des langages de programmation graphique. Les échanges et contributions se font et se feront dans les deux sens : de MUSICOLL vers cet enseignement, mais aussi en retour [12]. En effet, certains choix de conception d'objets dans Kiwi sont directement issus d'observations faites dans le cadre du cours lors des années passées : nous avons exposé au paragraphe précédent comment nous avons été amenés à poser qu'un argument écrit explicitement dans un objet ne soit plus modifiable de l'extérieur.

Un lot du projet MUSICOLL est consacré au design pédagogique du nouveau cours en faisant appel à Kiwi, à l'observation de son déroulement, au bilan de sa refonte et à de futures publications. Ce travail de recherche et de conception sera mené en 2017, pour une mise en œuvre début février 2018, puisque ce cours a lieu au deuxième semestre de l'année universitaire. Partant d'un patch vierge, guidé par l'enseignant, l'ensemble de la classe serait amené à construire collectivement le patch, par essais et erreurs, et à résoudre les éventuels problèmes posés par l'exercice, le tout étant ensuite archivé sur la plateforme Moodle[19] adoptée par notre université. L'émulation et la dynamique collective permettraient d'accélérer sans aucun doute le rythme de l'appropriation collective et d'aller bien plus loin dans l'expertise de la compétence.

Nous réfléchissons aux modalités pratiques de mise en œuvre du cours, notamment à l'adéquation entre l'organisation en classe et l'organisation sur support numérique : les étudiants et l'enseignant seront-ils tous connectés au même patch ? Plusieurs patchs répartis sur de petits groupes sont-ils préférables ?

6.2. Diffusion du projet Kiwi dans la communauté musicale

Nous souhaitons diffuser Kiwi auprès de la communauté musicale en l'associant à la création, la recherche et l'enseignement. Du point de vue de la création, nous confierons le logiciel à des compositeurs-chercheurs étudiants et enseignants à Paris 8, en les associant par leurs retours aux avancées du projet. Nous intégrerons leurs créations aux concerts produits par le CICM. Dans un second temps, Kiwi sera diffusé auprès de la communauté d'informatique musicale sous la forme non plus d'un logiciel *standalone* mais d'un *plugin* disponible pour l'environnement Ohm Studio. Enfin, nous imaginons que Kiwi pourra contribuer à de nouvelles pédagogies musicales, que ce soit en collège, en lycée, au conservatoire, ou même en ligne sous la forme de MOOC[20].

7. CONCLUSION

Après avoir dressé un bref état de l'art montrant l'émergence des pratiques collaboratives et nomades au sein de la création musicale audionumérique et la demande d'outils dédiés et adaptés, nous avons présenté le projet MUSICOLL. Nous avons mis en avant le contexte dans lequel il a été créé et défini ses axes majeurs de recherche et développement : la représentation de l'espace de travail et du temps dans les processus de travail collaboratif, la pérennité des réalisations et l'importance de l'apprentissage comme dimension fondamentale du collaboratif. Plus concrètement, nous avons présenté les possibilités offertes aux utilisateurs par la première maquette du logiciel de *patching* audio collaboratif Kiwi issu des travaux de recherche et développement du projet. Le développement d'une

[19] https://moodle.org/

[20] Massive Open Online Course

application collaborative telle que celle-ci a posé un certain nombre de problématiques sur un plan technique, fonctionnel et ergonomique que nous avons exposées et pour lesquelles nous avons présenté des solutions. Nous avons alors évoqué les prochains développements qui permettront d'offrir de nouvelles fonctionnalités aux utilisateurs telles qu'un meilleur contrôle du patch, une meilleure visualisation de l'activité collaborative au sein du réseau, ou encore la possibilité de se connecter à distance. Ils apporteront ainsi de nouvelles réponses concrètes aux questions posées par ce projet. Puis nous avons exposé certaines spécifications et attentes plus générales relatives aux logiciels de *patching* audio dans un environnement temps réel. Enfin, nous sommes revenus sur les différents contextes d'utilisation et de diffusion du logiciel que ce soit auprès de compositeurs-chercheurs étudiants et enseignants de l'université Paris 8 que dans le cadre d'un cours de Licence spécifique à son apprentissage dont la mise en place ouvre de nombreuses perspectives de recherche.

8. REMERCIEMENT

La majeure partie de la recherche exposée dans cet article est financée dans le cadre de la convention attributive d'aide de l'Agence Nationale de la Recherche n°ANR-15-CE38-0006-01.

9. REFERENCES

1. Allison, J., Oh, Y., Taylor, B., NEXUS: « Collaborative Performance for the Masses, Handling Instrument Interface Distribution through the Web ». Actes de la Conférence *New Interfaces for Music Expression 2013*, Daejeon, Corée du Sud, 2013.
2. Barbosa, A., « Computer-Supported Cooperative Work for Music Applications ». Thèse de doctorat, Université Pompeu Fabra, Barcelone, Espagne, 2006.
3. Colafrancesco, J., Guillot, P., Paris, E., Sèdes, A., Bonardi, A., « La bibliothèque HOA, bilan et perspectives ». Actes des *Journées d'informatique Musicale* (JIM), Saint-Denis, France, 2013.
4. Collins, N., McLean, A., Rohrhuber, J. & Ward, A., « Live Coding in Laptop Performance », *Organised Sound*, Vol. 8, Iss. 3, pp. 321-330, Cambridge, Royaume-Unis, 2003.
5. De Lavergne, C., Heïd, M.-C., « Former à et par la collaboration numérique », tic&société [En ligne], Vol. 7, N° 1 | 1er semestre 2013, mis en ligne le 4 juin 2013, consulté le 3 octobre 2016. URL : http://ticetsociete.revues.org/1308; DOI: 10.4000/ticetsociete.1308
6. Fencott, R., & Bryan-Kinns, N. « Hey Man, You're Invading my Personal Space! Privacy and Awareness in Collaborative Music ». Actes de la 10ème conférence *NIME*, pp. 198-203, Copenhague, Danemark, 2010.
7. Guillot, P., « Une nouvelle approche des objets graphiques et interfaces utilisateurs dans Pure Data », Actes des *Journées d'informatique Musicale* (JIM), Bourges, France, 2014.
8. Jourde, F., Laurillau, Y., Nigay, L., « Conception de systèmes collaboratifs multimodaux : analyse comparative de notations », Actes de la conférence *IHM*, Grenoble, France, 2009.
9. Laliberté, M., « Émergence et développement de l'informatique musicale ». *Théories de la composition musicale au XX° siècle*, Symétrie, Lyon, France, 2013.
10. Malloch, J., Sinclair, S., & Wanderley, M. M., « A network-based framework for collaborative development and performance of digital musical instruments ». Dans le symposium international *Computer Music Modeling and Retrieval*, pp. 401-425, Springer Berlin Heidelberg, Copenhague, Danemark, 2007.
11. Miletto E. M., Pimenta M.S., Bouchet F., Sansonnet J.-P., Keller, D., « Principles for Music Creation by Novices in Networked Music Environments ». *Journal of New Music Research*, Vol. 40, Iss. 3, 2011.
12. Sèdes, A., Bonardi, A., Paris, E., Millot, J., Guillot, P., « Teaching, investigating, creating: Musicoll », actes du colloque international *Innovative Tools and Methods for Teaching Music and Signal Processing*, sous la direction de L. Pottier, Presses des Mines, Paris, France, pp. 63-72, 2017.
13. Wang, G., Misra, A., Davidson, P., Cook, P., « CoAudicle: A Collaborative Audio Programming Space », Actes de la conférence *ICMC*, Barcelone, Espagne, 2005.