PÉDAGOGIE, RECHERCHE ET CRÉATION EN MUSIQUE AVEC ORDINATEUR AU DÉPARTEMENT DE MUSICOLOGIE DE L'UNIVERSITÉ JEAN-MONNET DE SAINT-ÉTIENNE

Laurent Pottier
CIEREC – Université Jean-Monnet
laurent.pottier@univ-st-etienne.fr

RÉSUMÉ

Tous les ans, à l'université Jean-Monnet (UJM) de Saint-Étienne, les étudiants en deuxième année de licence de musicologie suivent pendant deux semestres, en deux groupes, un enseignement obligatoire intitulé « Atelier d'informatique musicale » de 12 heures par semestre.

Ils y abordent dans un premier temps les bases d'un langage de programmation spécifique adapté au traitement du son en temps réel ou à la composition assistée par ordinateur (CAO). Selon les années, il peut s'agir des langages Max, Csound, Pure Data, Faust ou OpenMusic. Les étudiants apprennent à maîtriser des éléments de lutherie numérique pour construire des instruments de musique sur ordinateur – synthétiseurs et processeurs d'effets – qu'ils peuvent utiliser sur leurs propres ordinateurs ou smartphones. Ils composent collectivement – ou individuellement – une pièce de musique électronique ou mixte d'une dizaine de minutes, qu'ils doivent par la suite interpréter en direct, avec une présentation au public et/ou un enregistrement en studio en fin d'année.

Nous décrivons dans ce texte le travail réalisé au cours de l'année 2019-2020 dans ce cadre sous notre direction.

1. INTRODUCTION

Le département de musicologie de l'UJM [5] repose sur une équipe d'enseignants-chercheurs spécialisés respectivement dans trois domaines – en plus des disciplines musicologiques plus traditionnelles – : le jazz, l'ethnomusicologie et les musiques utilisant les technologies électroniques et numériques. Les étudiants sont donc formés à la musicologie en bénéficiant d'enseignements spécialisés dans ces trois domaines pendant les trois années de la licence [1].

Au niveau du troisième cycle, le master « Recherche » débouche sur le doctorat en musicologie, pouvant porter sur des thèmes variés allant de la musique du Moyen Âge à celle du XXIe siècle, en passant par les musiques du monde, les musiques traditionnelles, le jazz, les musiques électroacoustiques, les musiques électroniques et les musiques dites actuelles. Les enseignants du département encadrent également des thèses du doctorat « Recherche et Pratique » destiné à des interprètes et compositeurs, en partenariat avec le Conservatoire National Supérieur de Musique et de Danse (CNSMD) de Lyon [2].

Deux masters à vocation professionnelle sont aussi proposés à l'UJM : « Administration et Gestion de la Musique » (AGM) et « Réalisateur en Informatique Musicale » (RIM), une formation unique en France [3]. Le master RIM a été créé en 2011 et comprend des enseignements dans quatre axes principaux : a) techniques de studio et composition électroacoustique ; b) acoustique et traitement du signal ; c) informatique appliquée (CAO, temps réel) ; d) musicologie et gestion de projets. Les trois quarts des enseignements sont délivrés par des professionnels [4].

2. ENSEIGNER LA PROGRAMMATION INFORMATIQUE

Il est relativement difficile d'enseigner la programmation informatique à des étudiants de licence deuxième année en musicologie, sachant que la plupart ont suivi une formation littéraire durant leur cursus secondaire.

Dans les années quatre-vingt, le plan national « Informatique pour tous » (IPT) imposait, en première année de l'enseignement supérieur, toutes disciplines confondues, un nombre d'heures conséquent permettant aux étudiants d'aborder le fonctionnement d'un ordinateur, l'utilisation des outils de bureautique et comportait même une initiation à la programmation – par exemple le dessin avec la tortue du langage Logo ou l'utilisation du langage Hypertalk dans Hypercard pour la création de petites bases de données.

Ces vingt dernières années, les projets nationaux de ce type, malgré les annonces gouvernementales, témoignent d'ambitions beaucoup plus modestes. Dans la faculté Arts, Lettres, Langues de l'UJM, les enseignements en TICE (Technologies de l'Information et de la

[1] À titre d'information, la licence de musicologie regroupe une centaine d'étudiants répartis entre les trois années.

[2] Voir à ce sujet www.cnsmd-lyon.fr/fr-2/la-recherche/doctorants, accédé le 1/10/2020.

[3] Voir à ce sujet musinf.univ-st-etienne.fr/index.html, accédé le 1/10/2020

[4] Le Master RIM forme entre cinq et dix étudiants par an, dont un quart environ sont issus de l'UJM et trois quarts proviennent de l'extérieur.

Communication pour l'Enseignement) se limitent dans plusieurs départements à 4 heures de cours dans l'année, à peine de quoi survoler quelques fonctionnalités des logiciels de traitement de texte et de tableurs, et préciser les règles, droits et devoirs des internautes sur la toile.

Même s'il est clair que tous les étudiants n'utiliseront pas la programmation informatique dans leurs futures activités professionnelles, il semble toutefois utile voire fondamental de leur donner la possibilité et la chance d'aborder ce domaine afin de pouvoir susciter des vocations, mais aussi pour ne pas les laisser s'exclure face à ces outils omniprésents qu'il faut parfois être capable de détourner ou de personnaliser [6].

3. LES ATELIERS D'INFORMATIQUE MUSICALE EN L2 MUSICOLOGIE

Les salles de cours du département de musicologie de l'UJM comportent une vingtaine de postes de travail récents, équipés de claviers MIDI et de casques.

3.1. Apprentissage de Max

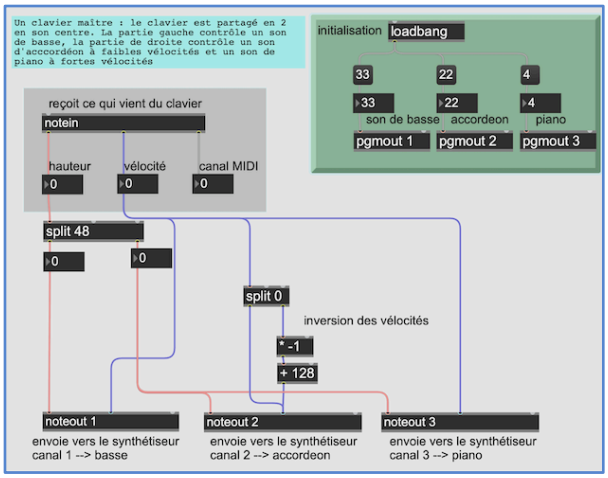

Figure 1. Premier patch MIDI dans *Max* : un clavier maître.

L'enseignement du langage Max débute par la présentation de la différence existant entre les données – et les types associés (*int, float, list, string*) – et les fonctions. L'apprentissage de la norme MIDI se fait en parallèle en demandant aux étudiants de programmer un clavier maître – zones de split, changements de programmes, transposition, courbes de vélocité (figure 1). Nous leur faisons ensuite aborder la question des algorithmes et des processus qui permettent de produire de la musique automatiquement. Cela passe par exemple par la combinaison des fonctions *metro* et *counter* pour la réalisation d'un chronomètre ou d'une gamme par tons (figure 2). L'enseignement se poursuit par la construction d'un séquenceur rudimentaire ou d'une boîte à rythme MIDI, sur le modèle de certains dispositifs analogiques ou numériques des années soixante-dix et quatre-vingt (figure 3).

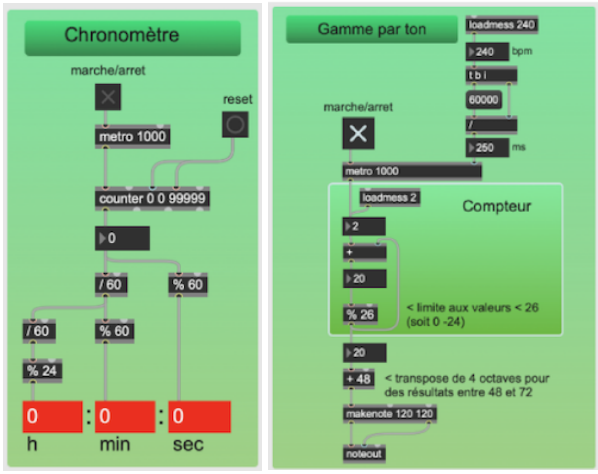

Figure 2. Combinaison d'un objet *metro* et d'un compteur dans *Max* pour créer un processus génératif.

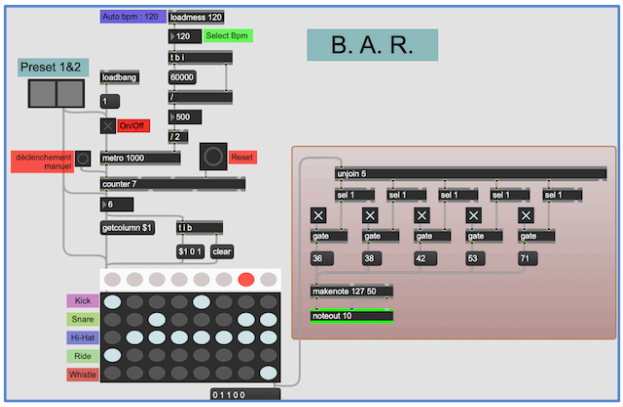

Figure 3. Une boite à rythme élémentaire dans *Max*.

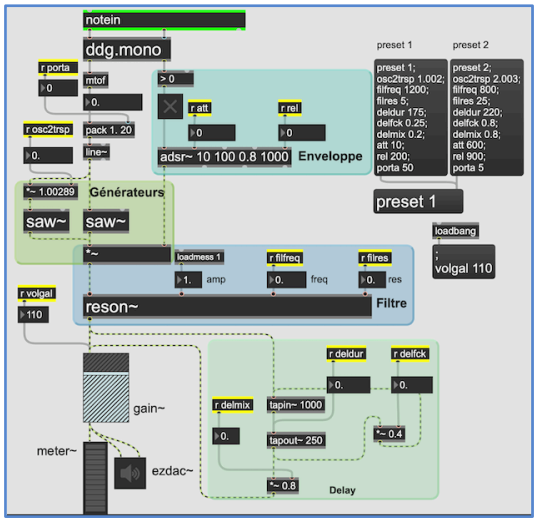

Figure 4. Un synthétiseur monophonique rudimentaire dans *Max*.

La deuxième étape consiste à faire découvrir aux étudiants les bases de la synthèse sonore et du traitement du signal. La liaison avec les exercices précédents se fait en reliant le séquenceur à un synthétiseur possédant les fonctionnalités principales qu'on pouvait trouver dès les années soixante et soixante-dix sur les synthétiseurs analogiques : plusieurs générateurs, avec des formes

d'ondes variées, une ou plusieurs enveloppes, un filtre et un delay (figure 4). À partir de là, les objets *poly* et *poly~* sont présentés pour permettre de rendre les instruments polyphoniques (figure 5). Un contrôle du *pitch-bend* et une modulation sont ajoutés.

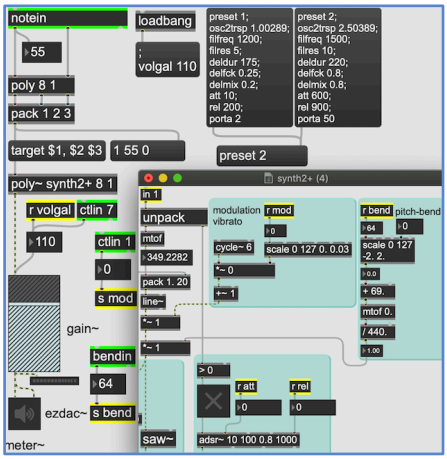

Figure 5. Un synthétiseur polyphonique dans *Max*.

3.2. Outils réalisés avec Max pour la production

Les enseignements se poursuivent en présentant la construction d'un sampleur polyphonique (figures 6 et 7) ainsi qu'un programme de traitement du signal, sorte de multi-effet programmable [5] (figures 8 et 9). Pour le multi-effet, deux *presets* ont été réalisés en modifiant simplement la matrice de câblage (figure 10). Pour le premier (à gauche), le son de l'instrument acoustique est envoyé à tous les effets simultanément et chaque effet est dirigé vers la sortie stéréo. La mise en route d'un effet se fait alors en montant le volume de sortie de l'effet. Pour le second *preset*, c'est le son du *delay* qui est envoyé à tous les effets. Sur un même ordinateur, il est possible d'ouvrir simultanément deux sampleurs indépendants et un multi-effet.

Figure 6. Un sampleur polyphonique dans *Max*. Sur le clavier du haut (*kslider*), les touches en bleu montrent les zones de *split*.

[5] Les programmes sont disponibles au téléchargement sur `mus-inf.univ-st-etienne.fr/recherches/synths.html`, accédé le 1/10/2020.

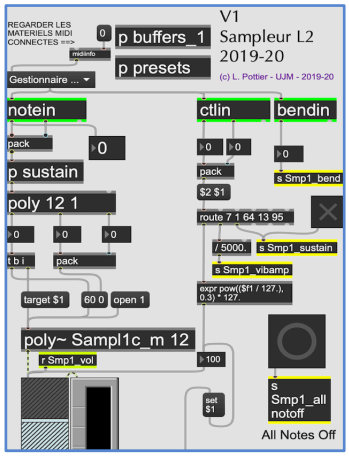

Figure 7. Le sampleur vu en mode édition (12 voix de polyphonie avec contrôle du *pitch-bend*, du volume, de la modulation et du *sustain*).

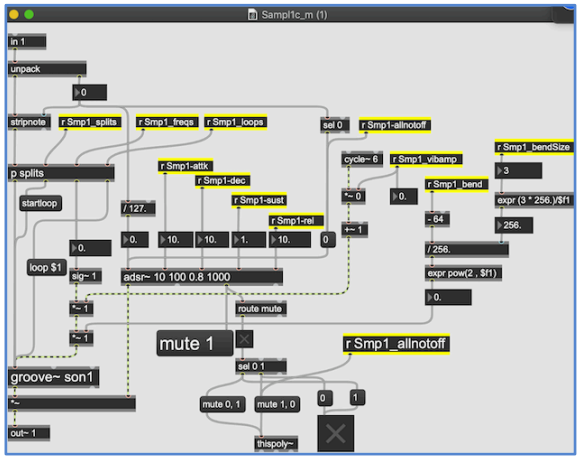

Figure 8. Le chœur du schéma DSP du sampleur dans *Max*.

Figure 9. Un multi-effet dans *Max* : modulation en anneau, *delay* (incluant une fonction *loop*), réverbération, harmoniseur (*gizmo~*) et granulateur (*munger~*).

Lors du second semestre, de janvier à avril, les étudiants composent une pièce d'une dizaine de minutes de façon collective avec les consignes suivantes : ils utilisent trois postes de travail avec un synthétiseur, un échantillonneur et un multi-effet. Toutes les notes jouées sont prélevées dans une partition fixée à l'avance et le paramètre principal à contrôler est le timbre global.

Chaque poste est équipé d'un ordinateur avec *Max*, d'une carte son professionnelle avec des entrées et sorties audio, de deux claviers MIDI, d'un mixeur MIDI, d'un microphone et d'une enceinte de monitoring. En option, ils peuvent disposer d'une pédale de *sustain*, d'une pédale de volume, de pads de percussion, de smartphones et d'une tablette graphique *multitouch*.

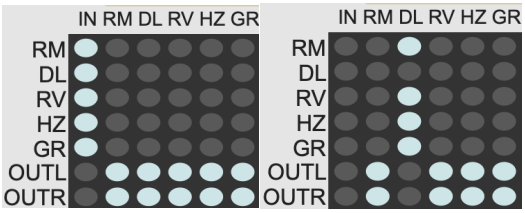

Figure 10. Les deux *presets* du patch de traitement au niveau de la matrice de câblage.

Figure 11. Les trois postes de travail dans une salle de répétition.

4. LES CRÉATIONS 2019-2020

En 2019-2020, le laboratoire CIEREC [6] a bénéficié d'un financement de l'IDEX de l'Université de Lyon (UdL) pour inviter deux compositeurs sous la forme d'une résidence à l'UJM, en partenariat avec GRAME [7]. Ces deux compositeurs, Vincent Carinola et Luis Quintana, ont encadré respectivement un groupe d'étudiants en Master 1 et un groupe d'étudiants en Licence 2 de musicologie pour leur faire produire chacun une pièce mixte interactive en vue d'une création lors du festival biennal *Musiques exploratoires* prévue le 22 mars 2020 au CNSMD de Lyon, en première partie d'un concert de l'Ensemble Orchestral Contemporain[8]. En raison de la situation sanitaire, ce concert a été reporté en février 2021.

Le projet présenté dans cette section concerne un troisième groupe d'étudiants, inscrits en Licence 2, que nous avons encadrés nous-même. La structure de la pièce a été imposée aux étudiants (figure 12).

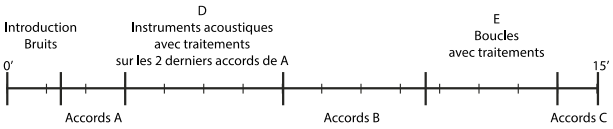

Figure 12. Structure de la pièce.

4.1. Introduction : bruits

L'introduction est jouée avec des smartphones sur lesquels ont été placées deux applications réalisées avec le langage Faust [9] [3]. Il s'agit d'un synthétiseur produisant un bruit blanc, disposant d'un filtre résonant et d'un système de granulation. Au départ de la pièce, huit étudiants produisent un bruit blanc filtré selon le dernier accord de la série A (figure 13). Progressivement, les filtres s'élargissent, faisant disparaître l'accord au profit du bruit blanc. Une seconde phase démarre alors, lors de laquelle le bruit blanc est progressivement fragmenté par le granulateur, se transformant en craquements. Une dernière phase débute ensuite, au cours de laquelle les smartphones sont relayés par les ordinateurs qui utilisent les mêmes instruments. Les sons produits, des sortes de craquements résultant d'une synthèse granulaire, traversent des filtres résonants dont les hauteurs correspondent au premier accord de la section A, ce qui permet une transition harmonique avec la section suivante. L'utilisation des ordinateurs avec des cartes audio permet d'augmenter la bande passante, notamment dans les fréquences graves.

[6] Centre Interdisciplinaire d'Études et de Recherches sur l'Expression Contemporaine. Voir `www.univ-st-etienne.fr/fr/cierec.html`, accédé le 1/10/2020.

[7] Centre National de Création Musicale, Lyon. Voir `www.grame.fr/`, accédé le 1/10/2020.

[8] Voir à ce sujet `www.grame.fr/evenements/2020-03-dans-l-affection-et-le-bruit-neuf`, accédé le 1/10/2020.

[9] Voir aussi section 4.4.

4.2. Sections A, B et C

Les sections A, B et C sont jouées par les étudiants sur les sampleurs pour construire des textures qui évoluent progressivement. Les notes à jouer sont à prendre dans un réservoir de hauteurs (figure 13). Les accords des sections A et B proviennent de la pièce *EnTrance* (1996) de Fausto Romitelli [2]. Les accords A sont ceux du finale de sa pièce, mais proposés ici dans l'ordre inverse – les accords sont étirés vers le bas et se contractent progressivement vers les aigus. Les accords B proviennent de la section 1B de la pièce originale, avec une alternance entre des accords de deux types différents, notés T et S. Les accords C ont été calculés avec *OM#* [10] par interpolation entre deux accords (figure 14).

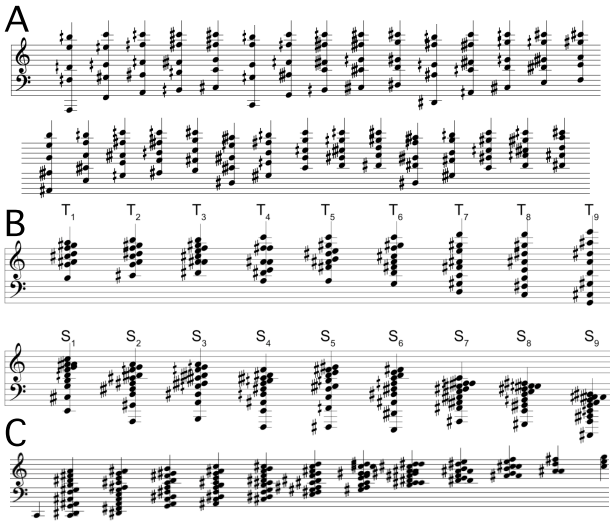

Figure 13. Les progressions harmoniques imposées des sections A, B et C.

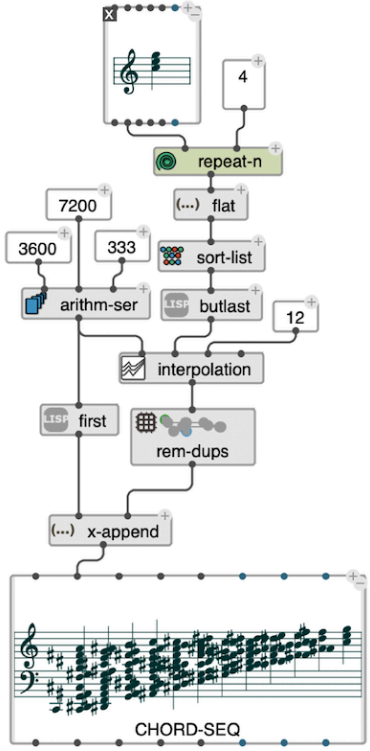

Figure 14. La progression harmonique C calculée dans *OM#*.

4.3. Sections D et E

La section D succède à la section A dont elle reprend les deux derniers accords. C'est une partie principalement instrumentale – avec piano, saxophone et trombone. Les sons produits sont envoyés dans le programme de traitement et les étudiants contrôlent des transformations du son grâce à des mixeurs MIDI qui permettent de gérer le volume de sortie de chaque effet et de modifier certains paramètres – fréquence du modulateur en anneau, taille et transpositions des grains du *munger* –, le clavier MIDI permettant de choisir les fréquences de l'harmoniseur. À la fin de la partie D, les étudiants capturent une boucle de chaque instrument acoustique – d'une durée comprise entre 3 et 5 secondes – qui sera ensuite utilisée dans la section E.

Dans la section E, les étudiants lisent en boucle les sons qui ont été préalablement enregistrés à la fin de la section D. Ces boucles sont envoyées dans l'ensemble des effets de patch *Max* pour y être traitées. À la fin de la section E, les boucles sont granulées avec une transposition qui descend progressivement vers les graves, ce qui permet de réaliser une transition avec le finale, la section C, qui démarre sur une note tenue grave – do_1 correspondant à 65,4 Hz. La section C part d'un accord grave dissonant pour progresser vers un accord final de *do* majeur.

4.4. Le synthétiseur Faust *WindMg*

Le synthétiseur Faust utilisé sur les smartphones comporte un générateur de bruit blanc (*no.noise*), un générateur d'enveloppe (*trigger*), une fonction temporelle aléatoire (*pulsar*) et un filtre (*moog_vcf_wind*). Le code (figure 15) peut être compilé pour smartphones Android à partir de l'éditeur Faust [11] du GRAME pour être ensuite directement téléchargé sur ces appareils à partir d'un QR-code affiché dans le navigateur (figure 16). Une autre version du synthétiseur a été réalisée, permettant un contrôle des paramètres via les trois accéléromètres des smartphones (figure 17). Cette version permet un contrôle gestuel du timbre plus interactif et expressif.

[10] *OM#* est le nouveau programme pour la composition musicale assistée par ordinateur de l'IRCAM et qui succède en janvier 2020 à *OpenMusic*. Voir à ce sujet `cac-t-u-s.github.io/om-sharp/`, accédé le 1/10/2020

[11] Voir à ce sujet `faust.grame.fr/tools/editor/`, accédé le 1/10/2020.

```
declare name        "WindMgCr3";
declare version     "1.03";
declare author      "LP - granul part adapted from sfIter by C. Lebreton";
declare license     "BSD";
declare copyright   "(c)CIEREC 2017-2020";
//----------------------- WindMgCr2 -----------------------
import("stdfaust.lib");
gain = hslider("[1]Gain", 0.2, 0, 1, 0.01):si.smooth(0.9997);
freq = hslider("[2]Freq", 440, 20, 5000, 1):si.smooth(0.9997);
res  = hslider("[3]Res", 0.8, 0, 0.995, 0.001):si.smooth(0.9997);
crounch  = hslider("[4]Crounch", 0.0, 0, 1, 0.01):si.smooth(0.9997);
freq2 = hslider("[5]Grain Size", 200,100,2205,1);
speed = hslider ("[6]Speed[unit:Hz]", 10,1,20,0.0001):si.smooth(0.9999);
proba = hslider ("[7]Proba[unit:%]", 70,50,100,1) * (0.01):si.smooth(0.9999);
P = freq2; // fundamental period in samples
Pmax = 4096; // maximum P (for delay-line allocation)
gain22 = 1;
//----------------------- FILTER -----------------------
moog_vcf_wind =  ve.moog_vcf(res,freq) * 0.5; // fix select normalized
//----------------------- NOISEBURST -----------------------
noiseburst = no.noise : *(gate : trigger(P))
    with {
    upfront(x) = (x-x') > 0;
    decay(n,x) = x - (x>0)/n;
    release(n) = + ~ decay(n);
    trigger(n) = upfront : release(n) : > (0.0);
    };
gate = phasor(1) :-(0.001):pulsar;
//----------------------- PHASOR -----------------------
phasor(init) =  (+(float(speed)/float(ma.SR)) : fmod(_,1.0)) ~ *(init);
//----------------------- PULSAR -----------------------
//Le pulsar permet de créer une 'pulsation' plus ou moins aléatoire (proba).
pulsar = _<:((_<(ratio_env)):@(100))*(proba>(_,abs(no.noise):ba.latch));
ratio_env = 0.5;
fade = (0.5); // min > 0 pour eviter division par 0
duree_env = 1/(speed: / (ratio_env*(0.25)*fade));
//----------------------- VU-METER -----------------------
tbar        = 0.1;                  // attack/release time in seconds
gbar        = exp(-1/(ma.SR*tbar)); // corresponding gain factor
envbar      = abs : *(1-gbar) : + ~ *(gbar) : ba.linear2db;
meter(x)    = attach(x, envbar(x) : hbargraph("level", -96, 10));

process= no.noise * (1 - crounch) + noiseburst * crounch * 2 :*(gain) :
    moog_vcf_wind :max(-0.99):min(0.99) : meter <: _,_;
```

Figure 15. Le code Faust du synthétiseur de bruit *WindMgC*.

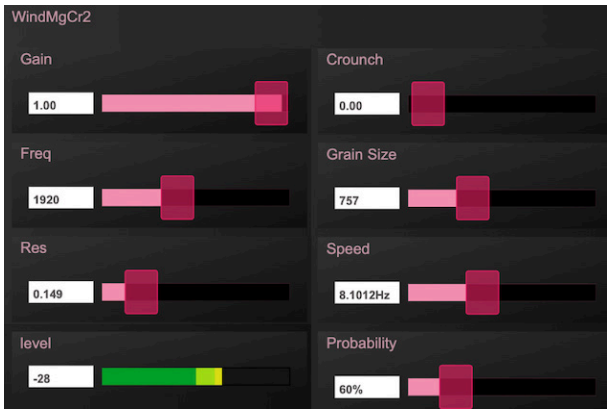

Figure 16. L'interface graphique du synthétiseur *WindMg*.

```
gain = hslider("[1]Gain[acc:0 0 -10 0 10]", 0.2, 0, 1, 0.01)
freq = hslider("[2]Freq[acc:1 0 -10 0 10]", 440, 100, 5000, 1
res  = hslider("[3]Res[acc:2 0 -10 0 10]", 0.8, 0, 0.995, 0.001)
crounch = hslider("[4]Crounch[acc:2 0 -10 0 10]", 0.0, 0, 1, 0.01)
freq2 = hslider("[5]Grain Size[acc:2 0 -10 0 10]", 200,100,2205,1)
speed = hslider ("[6]Speed[acc:0 0 -10 0 10]", 10,1,20,0.0001)
proba = hslider ("[7]Proba[acc:1 1 -10 0 10]", 70,50,100,1)
```

Figure 17. Les paramètres de contrôle par accéléromètres du synthétiseur *WindMgD*.

5. CONCLUSION

Le travail effectué avec des étudiants en licence de musicologie permet de les confronter avec le monde de la lutherie numérique, en leur montrant qu'avec des outils comme Max et Faust, il leur est possible de construire des instruments sur mesure dont ils peuvent ensuite profiter pour des pratiques créatives diverses.

L'objectif de ces compositions collectives consiste également à leur faire prendre conscience que la synthèse et le traitement permettent de créer et d'explorer de nouvelles sonorités, et qu'il est possible de faire de la musique en s'intéressant avant tout au timbre et à l'espace sonore en faisant abstraction, pour un temps, des musiques tonales calibrées qu'ils ont souvent tendance à pratiquer par ailleurs.

En raison de l'état d'urgence sanitaire n'ayant pas permis de travailler en présentiel pendant la seconde moitié du deuxième semestre, la pièce a été finalisée par les étudiants à distance, par le biais de réunions hebdomadaires en visioconférence. Pour cela, ils ont pu disposer d'une maquette réalisée dans le séquenceur *Reaper* qu'ils devaient compléter en ajoutant les sons créés avec Max ou Faust, sous la supervision d'une des étudiantes jouant le rôle de chef d'orchestre (figure 18). En dehors des séances de visioconférences, les étudiants ont donc dû faire fonctionner les outils sur leurs propres matériels et échanger entre eux pour produire des fichiers audio compatibles les uns avec les autres.

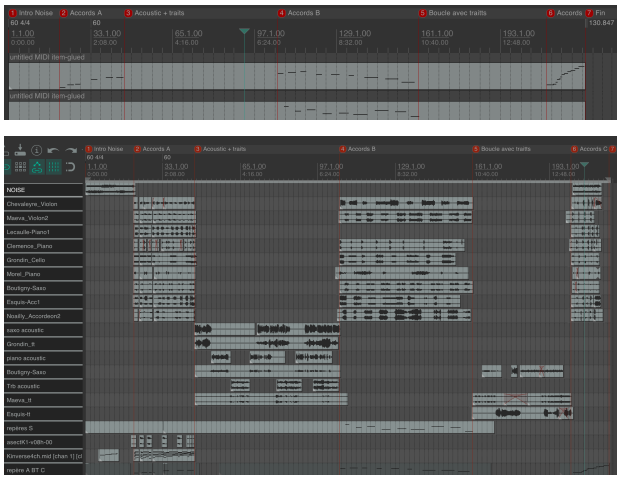

Figure 18. Maquette dans *Reaper* (en haut) et montage des sons produits par les étudiants (en bas).

L'initiation des étudiants en musicologie à la programmation informatique, doublée d'un apprentissage de techniques de traitement de signal, est toujours relativement délicate, mais le passage par un atelier de création en utilisant des outils qu'ils ont produits par eux-mêmes a tendance à les réconcilier avec les technologies (figure 19). Cette année, la crise sanitaire a conduit ces étudiants à installer les logiciels sur leurs propres équipements, ce qui n'a pas toujours été facile voire parfois impossible, même avec une assistance à distance. Mais ces ateliers ont permis de garder le contact avec les étudiants et entre les étudiants dans le cadre d'une activité donnée. Cela va nous inciter, pour les prochaines années, à insister plus fortement sur l'installation par les étudiants des logiciels étudiés sur leur propre matériel. Cela explique que nous insistions sur des logiciels multiplateformes, libres (Faust), à coût réduit (*Reaper*) ou utilisables en version démo « Runtime » (*Max*).

Pour finir, nous citerons deux expériences tout à fait novatrices portant sur la pédagogie des technologies avancées en informatiques musicale destinée à des musiciens : les travaux de l'équipe d'Alain Bonardi à l'Université Paris 8, avec les outils Kiwi pour une programmation partagée en ligne [4], et ceux de l'équipe de Jason Freeman de l'Université d'Atlanta avec la plateforme Earsketch [1].

6. RÉFÉRENCES

[1] Freeman, J., et coll. « Apprentissage avec EarSketch », *Revue Francophone d'Informatique et Musique* 6 (2018). [Dossier « Techniques et méthodes innovantes pour l'enseignement de la musique et du traitement de signal », Pottier L. (dir.), `revues.mshparisnord.fr:443/rfim/index.php?id=458`, accédé le 1/10/2020.]

[2] Olto, A., Pottier, L. « Analyse de *EnTrance* de Fausto Romitelli », `brahms.ircam.fr/analyses/`. [Mise en ligne prévue en novembre 2020.]

[3] Orlarey, Y., Fober D., Letz, S. « Syntactical and Semantical Aspects of Faust », *Soft Computing* 8 (2004), p. 623-632. [`doi.org/10.1007/s00500-004-0388-1`]

[4] Paris, E., Millot, J., Guillot, P., Bonardi, A., Sèdes, A. « Kiwi : vers un environnement de création musicale temps réel collaboratif – Premiers livrables du projet musicoll », Actes des Journées d'Informatique Musicale, Paris, 2017.

[5] Pottier, L. « Musique et musicologie à l'Université de Saint-Étienne : recherches en informatique musicale », Actes des Journées d'Informatique Musicale, Rennes, 2010.

[6] Pottier, L. « Quel intérêt pour des musiciens d'apprendre la programmation informatique ? », *Revue Francophone d'Informatique et Musique* 6 (2018). [Dossier « Techniques et méthodes innovantes pour l'enseignement de la musique et du traitement de signal », Pottier L. (dir.), `revues.mshparisnord.org/rfim/index.php?id=488`, accédé le 1/10/2020.]

Texte édité par Nathalie Hérold

Figure 19. Images des productions de l'atelier d'informatique musicale réalisées entre 2015 et 2020 (de haut en bas et de gauche à droite : 2015-2016 à l'UJM, 2016-2017 au FIL, 2017-2018 en répétition, 2019-2020 avant-première de la pièce de Luis Quintana au théâtre Tardy) © Images UJM L. Pottier et P. Landrivon.