La transmission de l'écriture du son : le cas de la synthèse sonore algorithmique

Antoine Villedieu

La distribution et le partage de données informatiques connaît actuellement un développement sans précédent. Dans ce contexte, l'apprentissage des programmes de synthèse sonore numérique[1] permettant la composition de musique électronique naît de plus en plus de l'échange d'innombrables ressources laissées en libre accès sur internet par les concepteurs de programmes et les compositeurs, ressources que d'autres compositeurs vont alors s'approprier pour créer leur propre musique. Un mode de transmission de synthèse sonore numérique très particulier voit le jour au cours des années 2000 : le partage de courtes lignes de codes destinées à l'environnement de programmation SuperCollider. Une fois le programme téléchargé, il suffit d'y copier ces lignes de codes puis de les exécuter pour en écouter le rendu sonore[2]. Ces lignes de codes sont alors modifiables à loisir, ouvrant la voie à une infinité d'autres lignes de codes pouvant également servir de base à l'élaboration de compositions électroniques de plus grande envergure (fig. 1).

```
r{inf.do{|i|Ndef(\,{VarSaw.ar(Duty.ar(1/12,0,Dseq((12..0)*(i%63+99)),
2)*[1,1.01],0,i/9%9/9)/9}).play.spawn;wait(1/3)}}.play
```

Fig. 1 : Code de Fredrik Olofsson posté sur son compte Twitter le 1er février 2013, 112e ScTweet du compositeur selon son classement[3]
© Fredrik Olofsson.

[1] Environnements tels que SuperCollider (https://supercollider.github.io), Max (https://cycling74.com/products/max) ou Pure Data (https://puredata.info). SuperCollider est un environnement de programmation en temps réel dédié à la synthèse sonore et la composition algorithmique conçu par James McCartney en 1996 et distribué librement depuis 2002 sous la Licence publique générale GNU (GPL-GNU). SuperCollider fonctionne sous tous les systèmes d'exploitation connus (Mac, Windows, Linux).

[2] Le langage de programmation de SuperCollider est *sclang*, son serveur audio est *scsynth*.

[3] Le compte Twitter de Fredrick Olofsson est accessible à l'adresse suivante : https://twitter.com/redFrik.

Les premières lignes de codes SuperCollider ont été conçues par le concepteur du programme lui-même, James McCartney. Très courtes en nombre de caractères, elles servaient d'exemples sonores et leur efficacité sonore était à l'image de la réussite du programme.

Les premiers compositeurs utilisant SuperCollider s'appropriaient ainsi ces courtes lignes, les modifiaient pour en créer d'autres, et voyaient dans cette pratique un moyen ludique et efficace d'apprendre à coder en SuperCollider. Certains d'entre eux ont alors commencé à partager leurs codes via une *mailing list*. Leur nombre commençait à grandir à mesure que les compositeurs se prenaient au jeu : construire la ligne de code donnant le résultat sonore et musical le plus impressionnant possible en un nombre de caractères aussi restreint que possible. Certains d'entre eux commençaient alors à partager leurs lignes de codes via Twitter[4]. Début août 2009, Dan Stowell, alors doctorant en informatique musicale à la Queen Mary University de Londres, lançait un appel à contribution auprès de compositeurs de *ScTweets*[5] pour constituer un recueil, qui sera publié en octobre 2009 par le magazine musical en ligne *The Wire*, recueil alors nommé Projet *SC140*[6]. Grâce à ce projet, le phénomène du *ScTweet* commençait à prendre de l'ampleur.

L'écriture du son

Un *ScTweet* donne à entendre une composition de synthèse sonore algorithmique minimale[7]. Son premier élément constitutif est le plus souvent un objet de production de signal audio, qui, une fois activé, produit une forme d'onde échantillonnée constituée de valeurs numériques lues successivement à partir d'une table d'onde. SuperCollider est pourvu d'outils permettant la visualisation d'une forme d'onde ainsi produite – en premier lieu l'outil *plot*.

Le deuxième élément constitutif d'un *ScTweet* est le plus souvent un algorithme variant le rendu sonore de cette forme d'onde. SuperCollider est pourvu d'un grand nombre d'objets qui, une fois combinés, permettent au compositeur de créer un algorithme de contrôle. Le compositeur peut évaluer à tout moment les valeurs produites par son algorithme – en les faisant apparaître à l'écran dans la fenêtre *Post Window* de SuperCollider prévue à cet effet – de manière à s'assurer

[4] Réseau social lancé en 2006 permettant l'envoi de messages limités au départ à 140 caractères.
[5] Tweets SuperCollider (lignes de code SuperCollider limitées en nombre de caractères).
[6] Les 22 codes du projet *SC140* sont présents à cette adresse sous forme de texte. Document en ligne consulté le 28 août 2023 : <https://ia800202.us.archive.org/29/items/sc140/sc140_sourcecode.txt>.
[7] Outre F. Olofsson, figurent parmi les compositeurs ayant composé des *ScTweets* Adam Armfield, Till Bovermann, Batuhan Bozkurt, Nick Collins, Thor Magnusson, Julian Rohrhuber, Dan Stowell, Andrea Valle ou Nathaniel Virgo.

de son bon fonctionnement et anticiper son résultat lorsqu'il sera appliqué au signal audio initial.

Le tout est maintenu dans une configuration minimale – troisième et dernier élément constitutif de ce type de code – du fait de sa restriction en nombre de caractères. Pour un compositeur de *ScTweets*, tout l'enjeu d'un code ainsi réduit à une seule ligne est de produire un résultat esthétiquement intéressant en jouant de cette forme de limitation.

L'analyse d'un *ScTweet* n'est pas chose aisée pour autant : un code nécessite toujours un décryptage, et son appropriation ne peut être immédiate car une bonne connaissance du programme informatique pour lequel il est écrit est requise, et un certain temps reste nécessaire pour en comprendre le fonctionnement. De plus, l'optimisation d'un *ScTweet* en nombre de caractères passe par des stratégies de codage rendant sa lecture plus difficile qu'un code de taille standard.

Mais les outils d'analyse, de visualisation et d'évaluation dont les environnements actuels de programmation disposent facilitent l'apprentissage de leur fonctionnement ainsi que l'appropriation des codes (s'il s'agit d'environnements textuels comme SuperCollider) ou des patches (s'il s'agit d'environnements graphiques tels que Max ou Pure data) destinés à y être lus et/ou modifiés.

De plus, quel que soit le type d'environnement de programmation utilisé, chacun de ses objets est pourvu d'un fichier d'aide (*Helpfile*) contenant généralement une série d'exemples sonores illustratifs, exécutables de manière à en écouter le rendu sonore. Les outils de visualisation, d'analyse et d'évaluation des programmes sont très présents dans leurs fichiers d'aide, et contribuent grandement à leur compréhension par le lecteur. La consultation du fichier d'aide d'un objet est essentielle pour en apprendre le fonctionnement et en faire une bonne utilisation.

L'écriture du son se situe dans le prolongement de la notation instrumentale, dont l'évolution historique témoigne de la volonté des compositeurs de maîtriser toujours plus précisément le rendu sonore de leur musique, même si ces deux types d'écriture sont de nature très différente, dans le sens où la notation musicale est essentiellement descriptive (une partition décrit un résultat à atteindre sans préciser complétement comment y parvenir) alors que l'écriture du son est essentiellement prescriptive (un code ou patch prescrit un ensemble d'instructions à la machine).

La synthèse sonore comme poursuite de la notation musicale : l'avènement du contrôle

La notation musicale occidentale, divisée en paramètres sonores distincts, s'est toujours attachée à contrôler certaines caractéristiques du rendu sonore instrumental ou vocal plus que d'autres, de façon inégale. L'ensemble des signes discrets d'une partition musicale – principalement des indications de hauteurs et de durées – est destiné à être matérialisé par le chant ou le geste d'un interprète sur son instrument de musique, ainsi le contrôle du rendu sonore que la notation musicale prescrit ne peut être que partiel, et la notation musicale « n'exerce son action que par rapport aux sous-entendus difficilement descriptibles de manière exhaustive que recèlent la pratique d'une époque et sa conception de l'interprétation[8] ». À ce sujet le philosophe Bernard Sève parle de « conventions sous-entendues, qui vont tellement de soi qu'il n'est pas nécessaire de les écrire. Elles font partie de la culture du milieu dans lequel et pour lequel la partition a été écrite, elles sont connues des exécutants[9] ». Ces conventions sous-entendues tendent progressivement à s'atténuer au cours de l'histoire pour faire place à une notation de plus en plus exhaustive, exerçant un contrôle de plus en plus grand sur le rendu sonore souhaité par le compositeur : la partition devient progressivement une référence à respecter le plus rigoureusement possible.

Certains compositeurs de musique instrumentale s'approprient à partir des années 1950 la synthèse sonore analogique dans le but de dépasser les limites intrinsèques de la notation musicale en matière de contrôle du phénomène sonore : le but était pour eux de prolonger leur pratique de la composition dans le son lui-même, de « composer le son » dans le prolongement direct de ce que la notation instrumentale leur offrait en matière de contrôle du phénomène sonore. La synthèse sonore permet selon le compositeur de musique électronique Jean-Claude Risset « d'étendre le contrôle compositionnel jusqu'à la microstructure, au niveau du son, ne pas se satisfaire d'un vocabulaire tout fait », de « composer le son même et créer à partir d'un matériau nouveau des formes nouvelles. Mettre directement en sons des structures ou des concepts. Dépasser la note, faire jouer le temps dans le son et pas seulement le son dans le temps. Composer les timbres comme des accords, prolonger l'harmonie dans le timbre[10] ». En somme, l'enjeu est de composer le son lui-même pour mieux le contrôler, et sont

[8] Jean-Yves Bosseur, *Du Son au signe : Histoire de la notation musicale*, Alternatives, 2005, p. 8.
[9] Bernard Sève, *L'Instrument de musique : une étude philosophique*, Le Seuil, 2013, p. 200.
[10] Jean-Claude Risset, « Synthèse et matériau musical », dans *La Synthèse sonore,* Danielle Cohen-Lévinas (dir.), Les cahiers de l'Ircam, coll. « Recherche et Musique », 1993, p. 55-56. J.-C. Risset (1938-2016) était un compositeur pionnier en musique électronique et un théoricien en informatique musicale.

concernées en premier lieu les caractéristiques du son que la notation musicale n'était pas de nature à pouvoir contrôler, à commencer par le timbre, c'est-à-dire la composition spectrale d'un son ainsi que son évolution temporelle.

Le son de synthèse numérique

Alors que la synthèse sonore analogique produit directement du son par la production de signaux électriques continus, c'est une forme d'écriture qui est à l'origine de la synthèse sonore numérique : une succession de valeurs est lue par la machine à partir d'une table d'onde, et constitue une courbe destinée à être convertie en signal électrique continu avant d'être transmis aux haut-parleurs. Une forme d'onde ainsi produite renferme toutes les caractéristiques du rendu sonore.

Un signal triangulaire numérique est à la base du code de Fredrik Olofsson pris comme exemple dans cet article : signal périodique (reproduisant le même cycle à l'identique indéfiniment), linéaire et continu, il est simplement constitué de deux courbes (l'une ascendante, l'autre descendante) constituées de valeurs numériques lues successivement d'une table d'onde. L'illustration centrale de la figure 5 est celle d'un cycle de signal triangulaire numérique produit par l'objet *VarSaw* de SuperCollider. Pour un taux d'échantillonnage de 48000 hertz (48000 valeurs numériques par seconde entre -1 et 1), une onde *VarSaw* paramétrée à 750 hertz produit 750 cycles identiques par seconde, chaque cycle contenant 64 valeurs successives (48000/750), dans cet ordre : 16 valeurs entre -1 et 0 ; 16 valeurs entre 0 et 1 ; 16 valeurs entre 1 et 0 ; 16 valeurs entre 0 et -1. Ce cycle est alors répété indéfiniment.

L'exécution du code suivant dans SuperCollider (fig. 2 et 3) permet d'obtenir la visualisation de ces valeurs.

```
{VarSaw.ar(750)}.loadToFloatArray(0.1,s,{|arr|a=arr});a.value.postln;
```

Fig. 2 : Code SuperCollider permettant la visualisation des premières valeurs numériques constituant la forme d'onde produite par l'objet VarSaw paramétré à 750 hertz.

```
-> FloatArray[ -1.0, -0.9375, -0.875, -0.8125, -0.75, -0.6875, -0.625
```

Fig. 3 : Début du résultat de l'évaluation du code ci-dessus dans SuperCollider.

Cette série de valeurs numériques est alors convertie en un courant électrique destiné à contrôler le mouvement d'une membrane de haut-parleur. Ce

processus est assuré par un convertisseur numérique-analogique : « cet appareil, contrôlé par une horloge d'échantillonnage, change les suites de nombres en séries de niveaux de tension[11] » qui constituent alors « une forme d'onde continue dans le temps [...], amplifiée, puis conduite vers le haut-parleur, dont les vibrations font changer la pression de l'air[12] ». La membrane de haut-parleur va alors créer du son, c'est-à-dire, d'un point de vue physique, une variation de pression d'air correspondant directement aux variations d'amplitude de la forme d'onde telle qu'elle apparaît à l'écran grâce aux outils de visualisation de forme d'onde dont les programmes disposent (fig. 4 et 5) :

> Un haut-parleur crée du son en se déplaçant d'avant en arrière, selon les changements de tension dans un signal électronique. Lorsque le haut-parleur « entre » par rapport à sa position de repos, la pression d'air diminue. Lorsque le haut-parleur « sort », la pression d'air près du haut-parleur augmente[13].

La synthèse sonore algorithmique

Un *ScTweet* calcule une forme d'onde échantillonnée à partir d'une formule mathématique : des fonctions élémentaires, mémorisées dans le code source de SuperCollider sous la forme de tables d'ondes, servent de base à son élaboration et sont progressivement combinées les unes aux autres lors du processus de codage : la composition d'un *ScTweet* se fait étape par étape, chaque nouvelle étape y apportant un nouvel élément constitutif, tel qu'un nouvel objet, ou l'ajout d'une valeur de paramétrage à un objet.

L'analyse d'un *ScTweet* consiste, à l'inverse, à décomposer ses éléments constitutifs de manière à comprendre comment ils s'articulent[14]. Elle est aussi interactive que sa composition : chaque niveau du code (délimité par des parenthèses) est un code en soi, il peut donc être immédiatement exécuté pour en évaluer le résultat sonore et/ou algorithmique, puis modifié, développé d'une autre manière dans une direction différente. « L'analyse révèle dans les œuvres des aspects insoupçonnés, elle les fait vivre, elle inspire d'autres œuvres », écrivait J.-C. Risset[15] : ces propos prennent un sens tout à fait littéral dans le cas

[11] Curtis Roads, *L'Audionumérique*, traduction de l'anglais par Jean de Reydellet, Dunot, 2016, p. 11.
[12] *Ibid.*, p. 11.
[13] *Ibid.*, p. 11.
[14] J.-C. Risset encourageant les compositeurs à l'analyse d'œuvres créées avec les moyens de l'informatique, écrivait en 1995 qu'il « serait dommage de voir les analystes se détourner des œuvres réalisées avec ordinateur – alors que l'ordinateur peut justement aider à fournir une représentation plus complète, pas seulement de l'œuvre achevée, mais aussi de sa génèse » (« Problèmes posés par l'analyse d'œuvres musicales dont la réalisation fait appel à l'informatique », *Analyse et création musicale*, Paris, L'Harmattan, 2001, p. 155).
[15] J.-C. Risset, *Analyse et création musicale*, *op. cit.*, p. 157.

de la pratique du *ScTweet* : de son analyse à son appropriation par le compositeur, il n'y a qu'un pas, franchissable immédiatement.

À la base du code de F. Olofsson pris comme exemple dans cet article se trouve un générateur de signal[16] permettant le passage progressif entre un signal triangulaire et un signal en dents de scie (normal ou inverse) : le cycle de la forme d'onde produite peut varier entre deux extrêmes, qui sont respectivement une rampe ascendante entre les valeurs numériques -1 et 1 (un cycle de forme d'onde dont le sommet est à droite, formant une onde en dents de scie) et une rampe descendante entre 1 et -1 (un cycle de forme d'onde dont le sommet est à gauche, formant une onde en dents de scie inversée). Le paramètre faisant varier la place du sommet de la forme du cycle s'appelle *width* : paramétré à 1, le sommet du cycle est à sa droite, paramétré à 0, il est à sa gauche, paramétré à 0.5, il est au centre du cycle (ce qui donne un signal triangulaire).

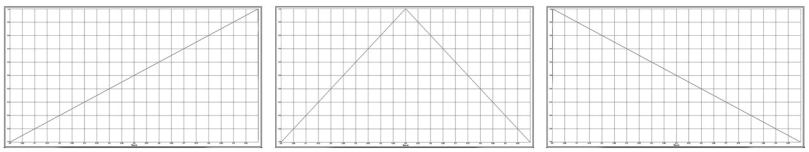

Fig. 4 : Les deux formes extrêmes d'un cycle d'une onde VarSaw (ici à gauche et à droite) visualisées à l'aide de l'outil « plot » de SuperCollider. Le paramétrage par défaut de l'onde VarSaw est l'intermédiaire entre ces deux extrêmes (ici au centre)[17].

La visualisation d'une forme d'onde numérique se produit d'une manière similaire d'un environnement de programmation à l'autre : ainsi, l'équivalent de l'objet *VarSaw* de SuperCollider dans l'environnement Max est l'objet *triangle~*. Max dispose de l'outil *scope~* permettant de visualiser les différentes variations de forme d'onde que peut produire l'objet *triangle~*.

[16] Il s'agit de l'objet VarSaw de SuperCollider. Sa fiche descriptive est disponible à l'adresse suivante : https://doc.sccode.org/Classes/VarSaw.html.

[17] Les trois codes SuperCollider permettant de visualiser 1 seconde de chacun de ces trois états de la *VarSaw* sont successivement les suivants : { VarSaw.ar(1,0,1) }.plot(1) ; { VarSaw.ar(1,0,0.5) }.plot(1) ; { VarSaw.ar(1,0,0) }.plot(1).

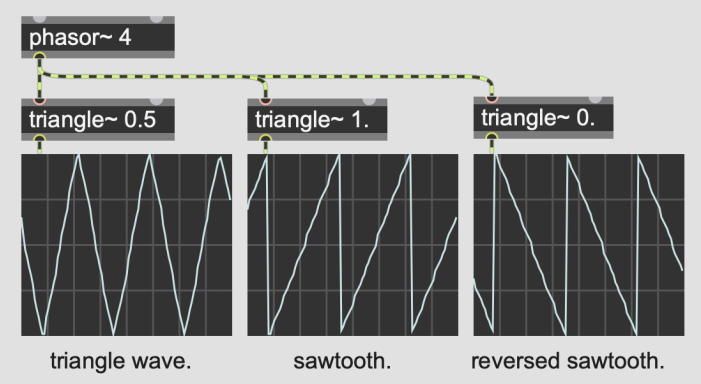

Fig. 5 : Illustration par l'outil scope de la visualisation des différentes formes d'onde que peut produire l'objet triangle dans l'environnement Max. Cette illustration provient de la fiche de documentation de l'objet triangle incluse dans le programme[18].

La variation progressive de la forme du cycle de la forme d'onde de synthèse générée par l'objet *VarSaw* produit un timbre dont la richesse en harmoniques varie dans le temps d'un extrême à l'autre. Un sonagramme permet de visualiser le contenu fréquentiel de ce changement de timbre dans le temps. Son axe vertical représente les fréquences (ici en kilohertz) et son axe horizontal le temps (ici en secondes) (fig. 6).

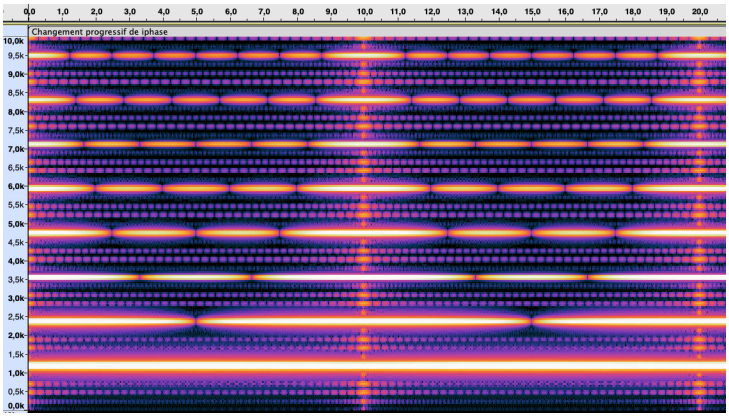

Fig. 6 : Sonagramme d'une onde VarSaw passant d'un timbre extrême à l'autre sur un intervalle de 10 secondes[19].

[18] La fiche de documentation de l'objet *triangle~* de Max est disponible à l'adresse suivante : https://docs.cycling74.com/max8/refpages/triangle~.

[19] Le code SuperCollider produisant le son à l'origine de ce sonagramme est le suivant : play{VarSaw.ar(1188,0,VarSaw.ar(1/10,0,0.5,0.25,0.25))}. Tous les sonagrammes de cet article ont été produits avec Audacity, logiciel d'édition audionumérique disponible à l'adresse suivante : https://audacity.fr. Les fichiers sons à l'origine de ces sonagrammes (fichiers mono convertis en stéréo) sont disponibles sur la plateforme HAL à l'adresse suivante : https://hal.science/hal-04330809.

Le spectre résultant du signal produit par l'objet *VarSaw* est plus riche en harmoniques lorsque son paramètre *width* est paramétré à 0 et 1 (ce qui correspond à 10 et 20 secondes dans le sonagramme de la figure 6). À l'inverse, il ne contient que des composantes harmoniques impaires lorsque le paramètre *width* est paramétré à 0.5 (à 5 et 15 secondes dans le sonagramme ci-dessus). La modulation progressive du paramètre *width* est ici assurée par l'ajout au code d'un autre objet *VarSaw* dont la lecture des valeurs numériques de la forme d'onde par la machine va produire une variation de timbre au signal initial, variation rendue automatique.

Pour reprendre le vocabulaire employé par J.-C. Risset, précédemment cité, en synthèse sonore numérique, le contrôle compositionnel du son par l'écriture s'étend désormais jusqu'à la microstructure du son.

Le *ScTweet* du 1er février 2013 de Fredrik Olofsson

Le code de F. Olofsson pris comme exemple dans cet article (fig. 1) est un exemple de composition algorithmique basée sur un processus automatique appliqué à la synthèse sonore : il consiste en premier lieu à contrôler de façon séparée les hauteurs et l'évolution du timbre d'un signal de synthèse généré par l'objet *VarSaw*.

Son premier élément musical est un arpège régulier descendant fondé de multiples entiers de 99 hertz[20]. Voici le sonagramme de cet arpège constitué de 12 fréquences descendant de 1188 à 99 hertz sur un espace d'une seconde, soit un douzième de seconde par fréquence (fig. 7).

[20] 99 * [12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1] = 1188, 1089, 990, 891, 792, 693, 594, 495, 396, 297, 198, 99 hertz. Ces fréquences concernent le canal gauche uniquement. L'exécution du code SuperCollider (12..0)*99 permet de visualiser ces valeurs dans la *Post Window*. Les fréquences du canal droit sont ces valeurs, toutes multipliées par 1.01, soit respectivement : 1199.88, 1099.89, 999.9, 899.91, 799.92, 699.93, 599.94, 499.95, 399.96, 299.97, 199.98, 99.99 hertz. Hormis ce léger décalage en fréquences entre les deux canaux produisant volontairement des interférences, la sortie sonore du canal droit est identique à celle du canal gauche. L'analyse de ce code dans cet article ne prend donc en compte que le canal gauche.

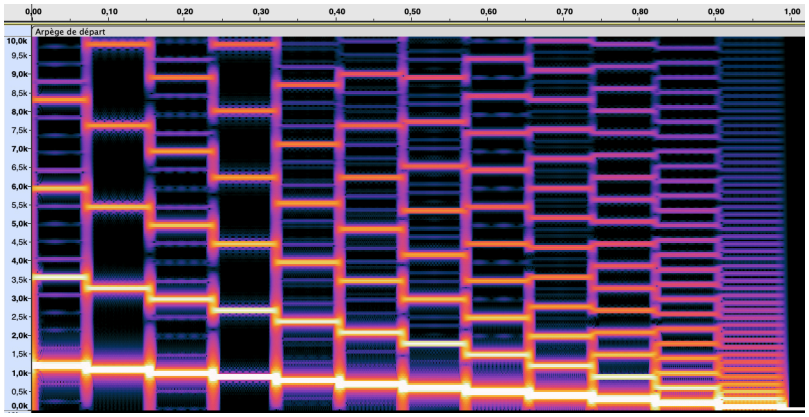

Fig. 7 : Arpège descendant à la base du code du 1er février 2013 de F. Olofsson[21].

Une nouvelle itération de cet arpège descendant est alors reproduite automatiquement tous les tiers de seconde : chaque itération durant une seconde, cela produit un tuilage entre les différentes itérations successives, tuilage visible dans le sonagramme suivant (fig. 8) : une fois les quatre premières fréquences de l'arpège descendant entendues, une deuxième itération de ce même arpège entre pendant que la première itération continue sa chute.

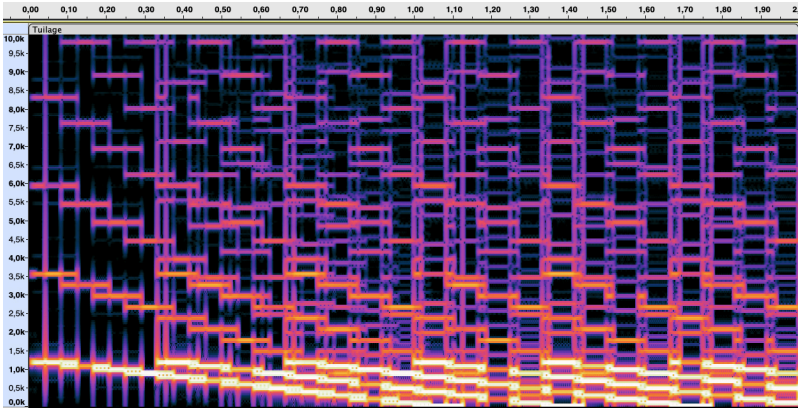

Fig. 8 : Tuilage entre itérations identiques de l'arpège initial sur un intervalle de 2 secondes[22].

Chaque nouvelle itération de l'arpège est alors haussée d'un hertz par rapport à l'itération précédente[23]. Le sonagramme suivant montre le tuilage produit par la réitération automatique de cet arpège à laquelle est ajoutée la progression ascendante de l'arpège d'une itération à l'autre (fig. 9).

[21] Sonagramme réalisé avec le paramétrage par défaut de la forme d'onde de l'objet VarSaw (width = 0.5). Le code SuperCollider produisant le son à l'origine de ce sonagramme est le suivant : {VarSaw.ar(Duty.ar(1/12,0,Dseq((12..0)*99)))}.play

[22] Le code SuperCollider permettant de produire le son à l'origine de ce sonagramme est le suivant : r{inf.do{|i|Ndef(\,{VarSaw.ar(Duty.ar(1/12,0,Dseq((12..0)*99),2))/9}).play.spawn;wait(1/3)}}.play

[23] 99 + [0, 1, 2, 3, 4, 5, ...] = 99, 100, 101, 102, 103, 104... hertz.

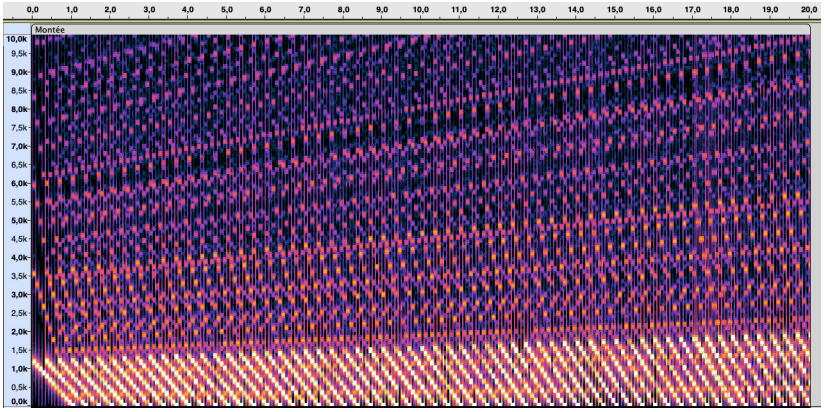

Fig. 9 : Montée d'un hertz à chaque nouvelle itération de l'arpège initial (tous les tiers de seconde) sur 20 secondes[24].

Cette montée d'un hertz de l'arpège initial à chaque nouvelle itération est alors automatiquement stoppée au bout de 63 itérations pour revenir aux fréquences de l'arpège initial (les 12 premiers multiples entiers d'une fréquence de 99 hertz). Une boucle infinie de 63 arpèges est alors mise en place (fig. 10)[25].

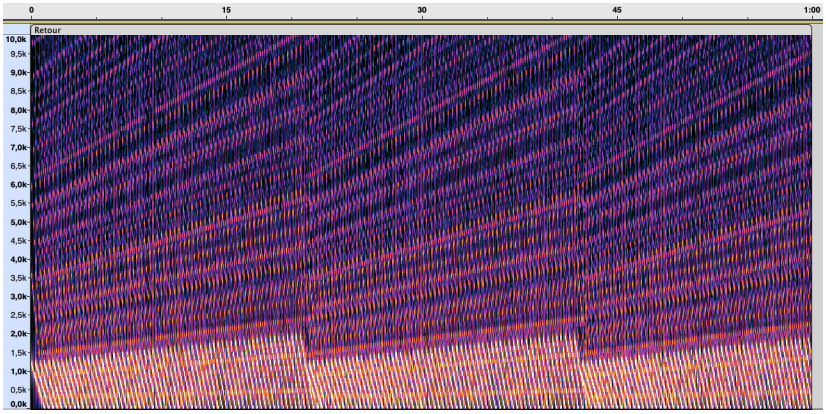

Fig. 10 : Retour périodique de l'arpège à une fondamentale de 99 hertz toutes les 63 itérations sur une minute[26].

De plus, le timbre de cette onde de synthèse varie progressivement dans le temps par la variation de la forme de son cycle de la manière décrite précédemment (fig. 11).

[24] Le code SuperCollider produisant le son à l'origine de ce sonagramme est le suivant : r{inf.do{|i|Ndef(\,{VarSaw.ar(Duty.ar(1/12,0,Dseq((12..0)*(i+99)),2))/9}).play.spawn;wait(1/3)}}.play

[25] Itération 0 = 99 hertz ; Itération 1 = 100 hertz ; (etc.), Itération 61 = 160 hertz ; Itération 62 = 161 hertz ; Itération 63 = 99 hertz ; Itération 64 = 100 hertz (etc.).

[26] Le code SuperCollider permettant de produire le son à l'origine de ce sonagramme est le suivant : r{inf.do{|i|Ndef(\,{VarSaw.ar(Duty.ar(1/12,0,Dseq((12..0)*(i%63+99)),2))/9}).play.spawn;wait(1/3)}}.play

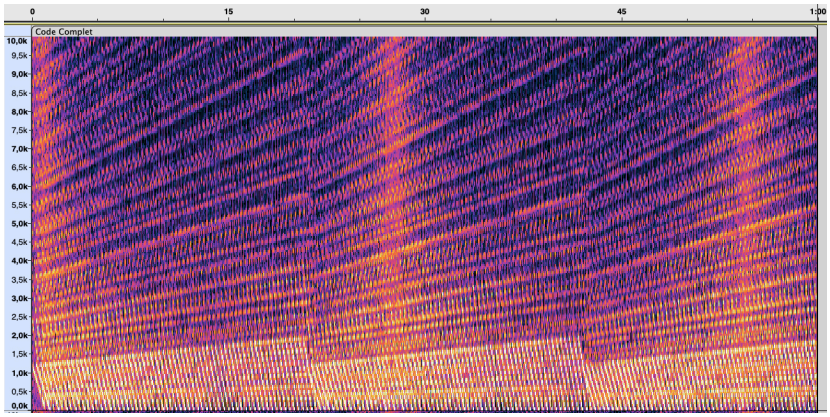

Fig. 11 : Sonagramme de la première minute du canal gauche du 112e code de F. Olofsson posté le 1er février 2013 sur son compte Twitter[27].

Chaque nouvelle étape de composition de ce code de F. Olofsson correspond à la fois à l'ajout d'un nouvel élément au code et à une échelle temporelle différente :

- Son arpège descendant de départ d'une durée d'une seconde est réitéré tous les tiers de seconde, ainsi tuilé à lui-même.

- Ses fréquences sont haussées d'un hertz à chacune de ses itérations jusqu'à revenir à son stade initial 63 itérations plus tard.

- Le timbre de cette onde varie d'un extrême à l'autre sur un cycle de 81 itérations.

Le dernier sonagramme de cet article rassemble les contenus de tous ceux qui le précèdent : c'est-à-dire le résultat de l'ensemble des éléments d'élaboration algorithmique de cette composition en une seule représentation visuelle.

L'économie esthétique

Un compositeur de *ScTweets* se donne pour but de faire en sorte que chaque étape compositionnelle de son code ait l'impact le plus clair et le plus efficace possible sur son rendu sonore final, puisqu'un *ScTweet*, du fait de sa concision, ne peut contenir qu'un nombre extrêmement restreint d'étapes compositionnelles, c'est-à-dire d'étapes d'élaboration algorithmique. Le but est de produire un résultat esthétiquement satisfaisant au sein de cette réduction en termes de moyens, qui est érigée en principe dans cette production musicale. Le phénomène du *ScTweet* se place ainsi sous le signe de l'économie esthétique.

[27] Le changement de *iphase* est organisé sur 81 itérations avant de revenir à son point de départ. Le code SuperCollider permettant de produire le son à l'origine de ce sonagramme est le suivant : r{inf.do{|i|Ndef(\,{VarSaw.ar(Duty.ar(1/12,0,Dseq((12..0)*(i%63+99)),2),0,i/9%9/9)/9}).play.spawn;wait(1/3)}}.play.

La question de l'économie des moyens a toujours préoccupé les artistes. « La faculté de bien me servir de mes moyens diminue lorsque leur nombre augmente », écrivait le cinéaste Robert Bresson[28]. Dans le même ordre d'idées, le compositeur Igor Stravinsky exprimait ainsi, en 1939, sa volonté de se limiter en termes de moyens :

> Plus l'Art est contrôlé, limité, travaillé et plus il est libre. [...]. Ma liberté consiste donc à me mouvoir dans le cadre étroit que je me suis à moi-même assigné pour chacune de mes entreprises. Je dirai plus : ma liberté sera d'autant plus grande et plus profonde que je limiterai plus étroitement mon champ d'action et que je m'entourerai de plus d'obstacles. Ce qui m'ôte une gêne m'ôte une force. Plus on s'impose de contraintes et plus on se libère de ces chaînes qui entravent l'esprit[29].

Le compositeur de musique électronique minimale Mark Fell explique dans un article de 2013 que les compositeurs ayant recours à la programmation, en théorie sans limites (hormis celle de la mémoire vive de l'ordinateur lorsque ces programmes fonctionnent en temps réel[30]) commencent généralement par en produire des configurations musicales réduites[31]. Ici encore, s'imposer des contraintes semble constituer l'élément déclencheur de créativité, dans le cas de la programmation de synthèse sonore algorithmique :

> Il est souvent dit que les programmes tels que Max/MSP sont des systèmes ouverts, contrairement aux systèmes fermés que sont les stations audionumériques ou les synthétiseurs logiciels, qui ont des fonctionnalités prédéfinies. Les systèmes ouverts, en revanche, offrent à l'utilisateur une sorte de toile vierge [...]. Et malgré ce qui peut être dit sur cette ouverture, la première chose que nous faisons lorsque nous nous approprions ces environnements prétendument ouverts est de développer des variations à partir de configurations extrêmement limitées [...]. Lorsque nous abordons cette ouverture en théorie infinie, nous commençons par nous construire des configurations fermées. Les utilisateurs de ces programmes peuvent dire qu'ils sont attirés par leur ouverture, mais ils sont en fait bien plus intéressés par les configurations plus réduites qui peuvent en naître[32].

La pérennité de l'écriture du son à l'épreuve de l'obsolescence technique

L'écriture du son est tributaire de programmes informatiques dont la survie dans le temps n'est pas garantie à long terme. Comment dans ces conditions peut-on croire en sa pérennité ?

[28] Robert Bresson, *Notes sur le cinématographe*, Gallimard, coll. « Folio », [1975], 1995, p. 9.

[29] Igor Stravinsky, *Poétique musicale*, Flammarion, coll. « Harmoniques », [1939], 2011, p. 105-106.

[30] Les environnements de programmation Max, Pure Data, et SuperCollider fonctionnent en temps réel.

[31] Mark Fell, « Collateral Damage », *The Wire*, 2013. Document en ligne consulté le 28 août 2023 : <https://www.thewire.co.uk/in-writing/essays/collateral-damage-mark-fell>.

[32] Traduction personnelle.

En effet, comme l'écrivait J.-C. Risset, « il est important de pouvoir décrire – écrire – les opérations effectuées sous une forme transmissible. Ce qui est en jeu, ce n'est pas seulement la commodité de l'analyse, mais la possibilité d'une écriture du son, et même – et surtout – la pérennité même de l'œuvre, qui risque de ne pas survivre à un dispositif technique éphémère[33] » : la question de la compréhension de l'écriture du son par l'analyste et/ou le compositeur, et par conséquent la possibilité de sa transmission, sont facilitées par les outils de visualisation, d'analyse et d'évaluation des programmes actuels.

Certes, chaque environnement a ses spécificités propres : interfaces graphiques différentes, manipulations algorithmiques plus ou moins aisées selon les programmes. Et les spécificités d'un programme donné influent grandement sur les décisions prises par le compositeur qui l'utilise : F. Olofsson a conçu ce code avec SuperCollider, dans la syntaxe SuperCollider. Il aurait certainement conçu une composition algorithmique très différente avec un environnement graphique tel que Max ou Pure data. Mais son code peut être traduit dans un environnement graphique et donner le même résultat sonore, du moins infiniment proche, ce qui peut être vérifié par le recours à un sonagramme : les formules mathématiques sur lesquelles se basent les tables d'ondes et algorithmes des objets à l'origine d'un son numérique sont invariantes d'un programme à l'autre, même si elles peuvent être écrites différemment dans leurs codes sources respectifs, ce qui explique les différences sonores infimes qui peuvent apparaître lorsqu'une telle composition a été transférée d'un programme à l'autre.

Qu'un compositeur fasse le choix d'un environnement graphique ou textuel, les différents environnements actuellement disponibles fonctionnent selon les mêmes modalités, et les outils de représentation visuelle, d'analyse et d'évaluation dont ils sont pourvus – similaires d'un programme à l'autre – facilitent la connaissance de leur fonctionnement en rendant explicite le contenu numérique d'un signal audio donné et le calcul sous-jacent à son contrôle algorithmique. Ces outils permettent par la même occasion la possibilité du transfert, de la réécriture d'une composition de synthèse sonore numérique d'un programme à l'autre.

[33] J.-C. Risset, *Analyse et création musicale*, *op. cit.*, p. 155.