Internship report

Computational analysis of jazz chord sequences

Romain VERSAEVEL, M1 Informatique Fondamentale, ENS de Lyon

Tutored by David MEREDITH, Associate Professor, Aalborg University,

leader of the Music Informatics and Cognition group

8 juin 2016

Table des matières

1 Introduction

2 Enjeux et présentation

Enjeux

C'EST QUOI L'INFORMATIQUE

ART ET SCIENCES, ART ET TECHNIQUE

#Citation(Breton, avant-propos)#

#Citation(Art+Science Now, p. 161)#

#Citation(Manoury, p. ? ? ?)#

3 Parcours de Karlheinz Essl

INTRO ! ! !

3.1 Formation : entre rock et éducation classique

Né le 15 août 1960 (? ? ?) à Vienne, Karlheinz Essl est issu d'un milieu aisé et cultivé. Son père, Karlheinz Essl senior, est un entrepreneur, qui a fait fortune en fondant la chaîne de magasins de bricolages bauMax [1]. Il est aussi connu comme collectionneur d'art, ayant acquis avec sa femme Agnes de nombreuses œuvres contemporaines à partir des années 1970, collection qui le conduit à faire construire son propre musée, le *Essl Museum*, dans la ville de Klosterneuburg située sur le Danube à quelques kilomètres au nord de Vienne.

Karlheinz junior est ainsi au contact d'un environnement culturel savant dès son plus jeune âge. Il étudie le piano dès l'âge de ? ? ? ans. Il explique cependant (cite ? ? ?) que cette expérience lui a déplu et qu'il n'a vraiment apprécié la musique que lorsque, adolescent, il s'est intéressé à la musique rock. Il prend alors part à plusieurs groupes

1. Entreprise dissoute en 2015, suite aux conséquences de la crise financière de 2007

et s'initie, en autodidacte, à la guitare électrique puis à la contrebasse. Néanmoins, ce n'est pas le rock « grand public » qui attire Essl, qui cite plus volontiers des groupes avant-gardistes ou expérimentaux comme *Gentle Giant* ou *Can*. C'est à travers ce dernier qu'Essl découvre la musique du compositeur allemand Karlheinz Stockhausen (deux membres de *Can*, le bassiste Holger Czukay et le claviériste Irmin Schmidt ont tous deux été ses élèves). Âgé de 15 ans, il achète un vinyle de Stockhausen, décrivant la découverte comme un véritable « choc » [2]. Cet engouement naissant pour l'électronique le conduira même à construire un synthétiseur ? ? ?pourri [3].

Lorsqu'il arrive dans les études supérieures, Essl décide (chimie ? ? ?) décide de se consacrer à la musique. Cette décision est prise à l'encontre de ses parents, qui le destinaient à prendre la succession de la direction de l'entreprise familiale (ce que fera finalement son frère cadet, Martin Essl). Reçu à la prestigieuse Académie de musique et des arts du spectacle de Vienne (*Universität für Musik und darstellende Kunst Wien*), il y étudie l'harmonie et le contrepoint avec Alfred Uhl, la contrebasse avec Heinrich Schneikart, et la composition auprès de Friedrich Cerha. Progressivement il va abandonner la contrebasse pour se consacrer exclusivement à l'analyse et à la composition, suivant en outre des cours de Dieter Kaufmann sur la musique électro-acoustique. Ce parcours académique le conduit à rédiger une thèse [4] sur la « pensée-synthèse » chez Anton Webern sous la direction du musicologue Hans Schneider, achevée en 1991.

Au cours de ses études, Essl fait deux rencontres décisives. D'abord, en 1985, il rend visite à son ami Gerhard Eckel en stage à l'Institut de Sonologie (*Instituut voor Sonologie*) de La Haye (Pays-Bas), l'un des premiers centres de recherche en musique électronique et musique informatique. C'est à cette occasion qu'il fait la connaissance, en partie par hasard, de Gottfried Michael Koenig, compositeur allemand pionnier de la musique algorithmique, qui dirige l'Institut depuis 1964. Essl trouve la partition d'un quatuor écrit par Koenig (*Streichquartett 1959*) et, ayant des difficultés à l'analyser, contacte celui-ci, qui lui apprend qu'il a utilisé des opérations de hasard dans sa

2. Dans **OMNIA IN OMNIBUS: Behind the Scenes ???** « Zuhause habe ich mir Kopfhörer aufgesetzt und mir die Platte angehört. Ich war total schockiert. Das war für mich damals ganz grauenhafte Musik, aber es hat mich gepackt. » — « À la maison, j'ai mis mon casque et écouté le vinyle. J'étais complètement sous le choc. À l'époque, je trouvais cette musique atroce, qui pourtant m'a emballé. »
3. Voir **R\leC {\"u}ckblick / Vorschau - Der Komponist Karlheinz Essl im Gespr\leC {\"a}ch mit Annelies K\leC {**
4. CITE ! ! !

composition. Les deux compositeurs correspondent et collaborent depuis, Essl ayant même été invité à contribuer à l'élaboration de *Projekt 3*, projet resté inachevé faisant suite aux *Projekt 1* et *Projekt 2*, premiers programmes pour la Composition Assistée par Ordinateur (? ? ?). Puis Essl rencontre John Cage, compositeur américain emblématique du XX^{e} siècle. Il connaît déjà son œuvre mais peut l'approcher en personne lors d'un concert donné en son honneur à Vienne en 1988. L'œuvre de Cage marque profondément celle d'Essl, qui lui rend d'ailleurs hommage dans deux de ses créations : *In The Cage* (1987) et *FontanaMixer* (2004).

Cette période de formation, qui s'achève avec jusqu'à la soutenance de thèse en 1991, voit se former le style d'Essl. On peut y identifier ses principales influences théoriques. D'abord, il y a le **sérialisme**[5], auquel l'a initié son professeur Friedrich Cerha[6]. Ce mouvement, fondé à Vienne par la seconde école de Vienne (Arnold Schönberg, Alban Berg et Anton Webern), élabore à l'origine des techniques d'inspiration arithmétique pour éradiquer systématiquement la tonalité des compositions ; de très nombreux compositeurs du XX^{e} siècle s'en réclament et l'on développé. (? ? ?en parler plus ?) Sur le plan esthétique, l'école sérielle va de pair avec la philosophie d'Adorno, lui-même compositeur, qui promeut la **Nouvelle Musique** (*Neue Musik*), marquée par l'injonction avant-gardiste. Cette théorie esthétique, présente dans toutes les disciplines de l'art contemporain, exige des artistes une innovation radicale permanente pour mériter ce statut[7]. Elle se manifeste dans l'intérêt qu'Essl portera à l'ordinateur et sa participation à l'avant-garde.
Les autres influences notables sont celles des trois compositeurs déjà cités : **Stockhausen**, **Koenig** et **Cage**. Tous trois ont été parmi les premiers musiciens à explorer les possibilités de l'*œuvre ouverte*[8] et du hasard, problématiques très présentes dans l'œuvre d'Essl. Koenig a même affirmé même que « la maîtrise du hasard par le com-

5. Pour plus de détails, voir par exemple **WODON !!!**

6. Dans **???** Essl précise qu'avant de commencer sa formation auprès de Cerha ses goûts le portaient plutôt vers des mouvances opposées comme le néo-clacissisme. #Citation(Elektronische Musik / Komposition / Improvisation - Karlheinz Essl im Gespräch mit Silvia Pagano)#

7. #Citation(Les théories de l'art, p. 59)# Pour plus de détails sur l'esthétique d'Adorno, voir **ESTH\leC {\'E}TIQUE** et Theodor W Adorno, « Philosophie der neuen Musik » (1949)

8. Telle que définie par Umberto Eco dans Umberto Eco, *Opera aperta : Forma e indeterminazione nelle poetiche contemparanee*, t. 3 (Tascabili Bompiani, 1962), voir partie **??** pour plus de ? ? ?)

positeur est un problème centrale de la musique actuelle »[9]. De Stockhausen, on retrouve l'intérêt porté au son et en particulier au son de synthèse, avec l'intégration de la musique électronique dans la musique savante. De Koenig, une paternité technique directe, puisqu'il a initié Essl à la musique algorithmique. De Cage, la recherche de formes radicalement nouvelles, en particulier l'événement, la performance. Dans **GERZSO ???** le compositeur Andrew Gerzso résume ainsi le terreau dans lequel Essl a pu développer son œuvre : #Citation(Andrew Gerzso, New computational paradigms for computer music, p. 1)#.
Enfin, la dernière influence notable d'Essl est d'ordre philosophique ; il s'agit du constructivisme radical, auquel le compositeur a même dédié un essai en 1992 (**RADIKAL???**). Cette épistémologie repose sur l'idée que la « connaissance » est une construction de l'esprit par rapport au monde, qui peut de fait varier d'un individu à l'autre, sans qu'il soit possible de parler de « vérité ». Essl résume ainsi l'application de ce courant de pensée à sa pratique artistique : « La musique ne se produit donc pas seulement dans la salle de concert, mais principalement dans la tête de l'auditeur. Savoir que le monde est construit par les perceptions individuelles plutôt que la simple illustration d'une réalité extérieure est lourd de conséquences pour la composition musicale. Cela fait en effet de l'auditeur un co-créateur. » #Citation(sine fine... Unendliche Musik)#

Bien que le rock ait joué un rôle décisif, à la fois en étant la source de la passion qui a décidé Essl à faire des études de musique, et en l'amenant à Stockhausen et à la musique électronique, son parcours est ? ? ?

3.2 Vers la musique algorithmique

On l'a vu, la rencontre avec Gottfried Michael Koenig a amené Essl à s'intéresser à la musique algorithmique. Cet intérêt d'abord théorique connaît par la suite de nombreux développements sur le plan pratique, qui se construit parallèlement à la relation entre Essl et l'ordinateur.

Ici, il convient de préciser ce que signifie ce terme de « musique algorithmique », et, pour commencer, ce qu'est un algorithme. Ce mot, forgé à partir du nom du mathéma-

9. #Citation(.)#

ticien perse du IX^{e} siècle Muhammad al-Khuwārizmī, est défini [10] par le Larousse 2016 comme un « ensemble de règles opératoires dont l'application permet de résoudre un problème énoncé au moyen d'un nombre fini d'opérations ». Dans ce sens, une recette de cuisine aussi bien que la méthode pour poser une multiplication sont des algorithmes. La composition algorithmique est alors simplement « la création d'algorithmes dont le produit a des conséquences musicales notables et utiles au compositeur, alant de la production et de l'exploration de matériaux musicaux à la génération d'œuvres complètes » [11]. Au sens large, la musique algorithmique possède donc une histoire séculaire [12], à laquelle participent par exemple tous les traités mettant en place des systèmes de règles pour l'organisation d'une composition, dont le plus ancien connu, intitulé *Musica enchiriadis* (d'auteur inconnu), remonte au IX^{e} siècle (? ? ? d'autres exemples, jeux de dés etc. ? ? ?). Mais c'est l'essor de la notion d'algorithme se fait conjointement avec l'avènement de l'informatique. Les ordinateurs sont en effet des machines capables d'appliquer n'importe quel algorithme de traitement de données et d'information ; ce sont en outre des machines *programmables*, autrement dit non-spécialisées, auxquelles leur utilisateur peut confier des tâches algorithmiques diverses à son gré ? ? ?. Dans le même temps naît l'idée de *composition algorithmique*, lorsque des compositeurs s'emparent pour la première fois des ordinateurs. En 1957 paraît l'*Illiac Suite*, première partition générée par un ordinateur (l'ILLIAC I) programmé par Lejaren Hiller et Leonard Isaacson, ainsi qu'un enregistrement des *Diamorphoses* de Iannis Xenakis, qui a utilisé un ordinateur pour l'assister dans des calculs de probabilité ; et en 1964, c'est *Projekt 1* de Koenig qui voit le jour.

Un exemple ? ? ?

C'est en 1984, qu'Essl utilise un ordinateur pour la première fois, par l'intermédiaire d'un ami, testeur pour un journal ; l'année suivante, il investit dans un micro-ordinateur Atari ST, qui sort cette année-là. Les années 1970 en 1980 correspondent à un changement radical dans les pratiques informatiques, qu'illustre cet achat. Alors qu'auparavant la logique de conception d'ordinateurs tendait vers des machines centralisées

10. Pour une caractérisation plus poussée, qui n'est pas nécessaire ici, se reporter à **knuth1998art**

11. #Citation(Introduction to Computer Music, p. 299)#

12. Cette remarque n'est pas seulement anecdotique : elle montre que l'histoire de la musique est imprégnée d'une culture algorithmique, qui explique que l'informatique ait trouvé un écho important dans cette discipline

de plus en plus puissantes, apparaissent les *micro-ordinateurs*, à usage individuels, qui vont progressivement s'imposer au détriment des premières. Le tableau 1 montre quelques modèles emblématiques de cette tendance [13]. Quant au tableau 2, il montre que l'Atari ST était significativement moins cher que ses principaux concurrents, ce qui est l'un des facteurs de son succès. L'acquisition de ce micro-ordinateur par Essl s'inscrit donc pleinement dans la tendance historique de cette période. Comme nous le verrons dans les parties suivantes (3.3 et 3.4), l'idéologie de liberté associée à cette individualisation de l'ordinateur [14] jouera aussi un rôle important dans la suite du parcours d'Essl.

Modèle	**Année**	**Nombre d'exemplaires vendus**
Programma 101	1965	! ! !
Intel MCS-4	1971	! ! !
Altair 8800	1975	! ! !
Apple II	1977	! ! !
IBM PC	1981	! ! !
Macintosh	1984	! ! !
Amiga 1000	1985	! ! !
Atari ST	1985	$\sim$ 50.000

FIGURE 1 – Quelques modèles de la naissance de la micro-informatique.

Modèle	Atari 1040 ST	Commodore Amiga	Apple Macintosh Plus	IBM PC AT
Prix	999$ (mono) ou 1199$ (couleur)	1795$	2195$	4675$

FIGURE 2 – Prix des micro-ordinateurs les plus répandus en 1985

Sur son Atari ST, Essl programme d'abord en BASIC (*Beginner's All-purpose Symbolic Instruction Code*), un langage de programmation impératif généraliste conçu pour sa simplicité d'utilisation. Il se tourne ensuite vers un langage beaucoup plus exo-

13. Harris, Neil (1985-11-11). "Atari Sales Underestimated". InfoWorld (letter). p. 18. Retrieved 8 January 2015 ! ! !

14. Décrite par ! ! ! : #Citation(Breton, intro chap 11)# #Citation(Breton, p. 206)#

tique ? ? ?, le LOGO [15]. Celui-ci a été développé à partir de la fin des années 1960, à l'origine par deux chercheurs du MIT, Seymour Papert et Marvin Minsky. Les programmes écrits en LOGO dirigent le déplacement d'un curseur à l'apparence de tortue ; leur résultat est un dessin, celui de la trajectoire de la tortue. Cet apparence ludique est dû au fait que le LOGO, conçu en s'appuyant sur les théories de Jean Piaget, était destiné à permettre à des enfants de se familiariser avec la programmation ; il s'agit cependant d'un langage très complet qui permet par exemple de manipuler des fichiers ou des structures de listes. C'est principalement cette dernière fonctionnalité qui retient l'attention d'Essl, lequel implémente des procédures issues du sérialisme.

La pensée algorithmique devient prépondérante dans les œuvres du jeune compositeur. Sa première pièce ayant nécessité un programme informatique, *BWV 1007a* paraît en 1986 (en collaboration avec son ami Eckel). Ce programme a réalisé une analyse statistique de la *Première suite pour violoncelle* de Jean-Sébastien Bach (BWV 1007), afin d'en réaliser un découpage et une recombinaison mélodiquement cohérente. Précisons que cette expression peut être trompeuse : lorsque l'on parle de programmes et d'algorithmes, il est facile de laisser imaginer une « volonté » ou une forme de conscience de l'ordinateur. Il n'en est évidemment rien, le programme d'Essl ne fait en réalité qu'exécuter des instructions prévues par lui. Une recombinaison « mélodiquement cohérente » signifie ainsi simplement une recombinaison respectant des règles définies par Essl dans le but d'éviter des combinaisons sonores qu'il juge incohérentes. En cela, *BWV 1007a* est un bon exemple de ce qu'est — et donc de ce que n'est pas — un algorithme : Essl aurait théoriquement pu appliquer lui-même sa méthode, découper et recombiner la pièce de Bach, en s'assurant de respecter les règles mélodiques qu'il a prévues ; mais cette tâche aurait été particulièrement fastidieuse ; l'ordinateur permet de l'automatiser et de la réaliser plus rapidement. La même année, Essl compose l'une de ses premières œuvres majeures, le quatuor à corde *Helix 1.0*, qui lui vaut deux prix en 1987, au concours international de quatuors à cordes de Budapest et au concours de quatuors à cordes du Konzerthaus de Vienne. Cette pièce construite sur des structures élaborées représentant des mouvements en hélice (comme combinaison de deux mouvements, l'un circulaire, l'autre rectiligne) reflète

15. Plus précisément le xLOGO, l'une des nombreuses implémentations du langage original. Pour plus d'informations sur le LOGO, voir **harvey1985computer**

une véritable pensée algorithmique [16]. Les algorithmes impliqués ont cependant été réalisés sans l'aide de l'ordinateur.

Pour un compositeur qui a accordé aux aspects algorithmiques de la composition une place prépondérante dans son œuvre, Essl a une relation inhabituelle aux algorithmes. Sa position est résumée par cet échange dans une interview de 2015 : « Magdalena Halay : Qu'est-ce qui pour vous fait qu'une composition sonne algorithmique ? — Karlheinz Essl : Oh, le but n'est pas que mes compositions sonnent algorithmiques ! [. . .] Le but est qu'elles ne sonnent pas algorithmiques. » [17]. Les méthodes formelles impliquées dans le processus de composition d'Essl sont *théorisées*, *décrites* (dans des articles, des programmes accompagnant les performances ou leurs enregistrements, les cours et les interviews donnés par Essl. . .) ; mais elles ne sont pas *audibles*, ou du moins de sont pas mises au premier plan lors de l'écoute. Cette attitude s'oppose à celle de beaucoup d'artistes contemporains ayant recours à des algorithmes, qui au contraire en font la monstration à travers leurs œuvres [18]. Elle opère ainsi une sorte de synthèse entre l'esthétique de la virtuosité (du compositeur) et celle qui met les moyens de composition au service de l'expérience de l'auditeur.

Les dernières années d'études d'Essl, qui sont aussi ses premières années en tant que compositeur, marquent ainsi l'apparition et le développement dans sa pensée artistique de pratiques algorithmiques. Ce phénomène va de pair avec celui de l'avènement de la micro-informatique, qui permet à Essl d'expérimenter avec sa propre machine. Il élabore alors une articulation relativement originale entre le procédé algorithmique et le résultat esthétique qu'il élabore alors. Comme nous allons le voir dans la partie suivante, les progrès technologiques vont lui permettre d'enrichir encore grandement cette originalité.

16. Encore une fois, il y a deux causes à cette présente structurante forte des algorithmes : l'influence de la programmation informatique qui occupe Essl à cette période, mais aussi l'omniprésence de procédés d'ordre algorithmique dans la tradition musicale.

17. #Citation(Composing with Algorithms - A talk between Karlheinz Essl and Magdalena Halay)#

18. On peut trouver de nombreux exemples dans Stephen Wilson, *Art+Science Now* (Thames & Hudson, 2012), en particulier au chapitre 7 intitulé « *Algorithmes* », p. 160-179.

3.3 IRCAM et Composition en temps réel

Un tournant décisif dans l'histoire de la musique informatisée et dans la carrière d'Essl a lieu à la fin des années 1980 et au début des années 1990. Il s'agit de l'apparition de dispositifs en « temps réel »[19]. Ceux-ci offrent pour la première fois la possibilité d'une interaction immédiate entre l'utilisateur et une machine *informatique*[20] qui convertir le résultat de ses calculs en événements musicaux significatifs, typiquement sous forme de sortie MIDI (ce qui permet de produire du son avec des instruments comme le Yamaha Disklavier[21]) ou directement dans un format audio (écouté *via* des haut-parleurs). ! ! !http ://mustudio.fr/

Essl est confronté avec ces dispositifs en temps réel pour la première fois en 1992. Il est alors invité à l'Institut de recherche et coordination acoustique/musique (IRCAM) à Paris, institut public de création musicale et de recherche appliquée à la musique, qui lui passe commande d'une pièce pour ensemble et sa toute récente *Station d'informatique musicale* (SIM)[22]. Cette machine succède à plusieurs autres prototypes développés par l'ingénieur et physicien italien Giuseppe di Giugno depuis 1975, intitulés 4A, 4B, 4C et 4X. Essl non seulement compose la pièce qui lui a été commandée (*Entsagung*), mais s'initie au langage de programmation MAX/MSP avec lequel fonctionne la SIM, tirant profit de son expérience avec LOGO pour réaliser des expériences de composition. Le « temps réel » de l'IRCAM entraîne naturellement un gain de temps considérable, à travers deux ruptures. D'une part, les calculs sont exécutés plus rapidement : alors que les résultats des algorithmes de la bibliothèque *COMPOSE* nécessitaient parfois une nuit entière de calcul, la SIM les calcule immédiatement. D'autre part, ces résultats sont audibles, c'est-à-dire dans un format directement intelligible ; lorsqu'il programmait en LOGO, Essl n'obtenait que des résultats sous forme symbolique (par exemple des nombres représentant des hauteurs de notes et leurs durées), qu'il devait ensuite traduire en partition et interpréter sur un instrument traditionnel.

La découverte des machines en « temps réel » et de MAX/MSP est déterminante pour

19. Cette notion est discutée plus en détail partie **??**

20. La nouveauté réside dans le fait que c'est avec un ordinateur, une *machine à calculer*, que le temps réel est obtenu : en musique électronique, les synthétiseurs permettaient ce temps réel depuis plusieurs décennies. . . et les instruments de musique traditionnels, depuis toujours.

21. Piano mécanisé créé en 1987 ; la norme MIDI a quant a elle été établie en 1983.

22. Voir **lippe1991ircam** pour une description détaillée de la station.

Essl, qui adopte les immédiatement comme des outils de travail de premier plan. Il transporte dans ce nouveau langage les algorithmes de *COMPOSE* dans une bibliothèque intitulée *Realtime Composition library* (*RTC-lib*), qu'il continue à enrichir et à optimiser. Il compose de plusieurs pièces interactives en temps réel, parmi lesquelles son œuvre la plus célèbre, la *Lexikon-Sonate*. La *RTC-lib* et la *Lexikon-Sonate* font l'objet respectivement des sections **??** et **??**.

L'interaction musicale en temps réel n'est cependant possible qu'avec les machines prototypiques conçues par des centres de recherche comme l'IRCAM, qui n'existaient donc qu'en nombre très limité. L'accès y est de fait difficile et concurrentiel ; en outre, Essl souhaite composer en étant autonome de telles institutions. Il lui faut pour cela attendre 1998 et la sortie du premier iMac, suffisamment puissant pour obtenir des performances similaires à celle de la Station d'informatique musicale. Comme le montre la figure 3.3, cela coïncide avec un forte augmentation de sa productivité (en tant que nombre d'œuvres publiées).

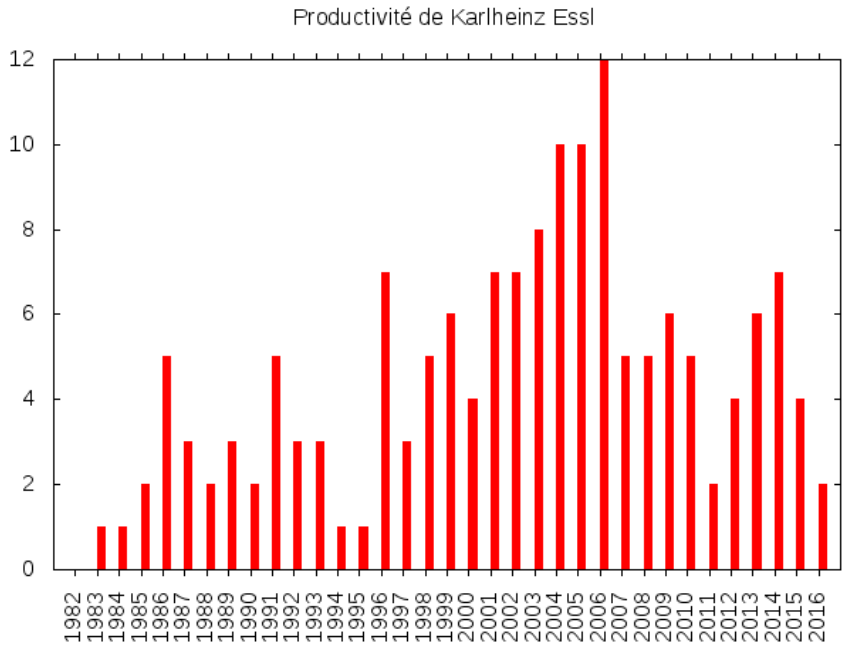

FIGURE 3 – !!!

Ces œuvres d'une forme inédite sont le volet pratique du développement d'Essl, qui formule aussi de nouvelles notions sur le plan théorique, le conduisant à son rapport actuel à la musique algorithmique. Il élabore en particulier deux notions étroitement

liées : celle de « composition en temps réel » (*Echtzeit Komposition*) et celle de « ? ? ? » *Strukturgenerator*.

Voici, pour les appréhender, deux définitions qu'Essl donne aujourd'hui des algorithmes : « [Les algorithmes] sont le fondement de ma pensée musicale, qui consiste à ne pas voir la musique seulement comme une expérience sensible, mais comme quelque chose qui comporte une multitude de structures plus profondes que l'on peut exprimer sous forme de modèles. » [23], et : « Il y a la définition classique, qui provient plutôt des sciences de l'ingénieur, selon laquelle un algorithme est une sorte de recette de cuisine pour résoudre rapidement un problème. C'est une approche possible, mais il y en a une autre que je trouve plus intéressante. Elle consiste à envisager l'algorithme comme la définition d'un méta-modèle, duquel on peut obtenir différents résultats en modifiant les paramètres du système. C'est en ce sens que j'utilise le terme d'algorithme, et c'est ainsi que je le comprends dans les travaux de Gottfried Michael Koenig et Karlheinz Stockhausen. ». [24].

Essl fait aussi plusieurs fois référence à la notion d'*Urpflanze* de Goethe, une plante originaire imaginaire dont on pourrait dériver toutes les espèces plantes [25]. Le mot de *modèle* est crucial dans l'idée de *Strukturgenerator* [26]. Le produit des algorithmes que conçoit Essl possèdent une structure imposée commune, mais varient selon les paramètres fournis en entrée et le hasard [27] interne des opérations exécutées par ces algorithmes. Le module *Triller* de la *Lexikon-Sonate*, étudié dans la partie **??**, est un exemple simple de *Strukturgenerator*. Tout ce que produit ce programme a une *structure* de trille : un ensemble de quelques notes répétées à grande vitesse ; mais les trilles générées par différentes utilisations *varient* par le nombre de notes différentes jouées, leurs hauteurs, la vitesse, la durée du trille… L'idée de définir des éléments musicaux comme des représentants d'un ensemble des possibles gouverné par une structure, sous la forme d'un système de règles, n'est toutefois ni nouvelle ni originale. Les règles strictes gouvernant le contrepoint classique relèvent exactement de ce cas

23. #Citation(OMNIA IN OMNIBUS : Behind the Scenes - Karlheinz Essl im Gespräch mit Katharina Hötzenecker)#
24. #Citation(Intuition, Automation, Entscheidung. Der Komponist im Prozess algorithmischer Komposition)#
25. **goethe1993italienische**
26. **Strukturgeneratoren - Algorithmische Komposition in Echtzeit**
27. Le rôle du hasard dans la composition en temps réel fait l'objet de la partie **??**.

de figure ; la seule nouveauté dans les *Strukturgeneratoren* d'Essl est qu'ils sont *actifs*, l'ordinateur permettant de créer la musique à la demande une fois le système de règles explicité et traduit en code informatique. On retrouve des procédés similaires dans nombre de musiques algorithmiques où le temps réel n'est pas nécessaire, et beaucoup de domaines usent de tels schémas, par exemple dans les jeux vidéos où il est aujourd'hui monnaie courante de générer ainsi des *terrains*.
En revanche, l'originalité d'Essl réside dans la manière d'utiliser ces *Strukturgeneratoren* : ce qu'il nomme la « composition en temps réel ». Celle-ci peut être décrite comme la conjugaison de la composition algorithmique avec le temps réel. Les pièces de l'œuvre d'Essl répondant à cette appellation peuvent être réparties entre deux catégories : les *installations* et les *méta-instruments*. Les installations sont des programmes qui génèrent de la musique sans intervention humaine, généralement utilisées dans un lieu précis comme à l'intérieur d'un musée. À la différence d'un simple enregistrement, le son diffusé par un programme de type installation est élaboré au fur et à mesure. Ses caractéristiques notables sont en particulier l'absence de répétition, mais aussi de début et de fin en tant que tels. Les *méta-instruments* sont des programmes offrant une interaction limitée (avec un *instrumentiste*), mais dont le produit est du même type que celui d'une installation ; c'est en quelque sorte une installation dont les interactive dont un petit nombre de paramètres internes peut être modifié en temps réel. Un instrument traditionnel comme le violon laisse une liberté à celui qui en joue (hauteur, durée, volume des notes…), mais impose aussi certains paramètres du son (timbre, ambitus, nombre de notes pouvant être jouées simultanément…). C'est ce qui justifie cette appellation de méta-instrument, la principale différence étant que les instruments traditionnels sont déterministes (le même geste produira systématiquement le même son) mais généralement pas les méta-instruments programmés par Essl. Comme nous le verrons par exemple à travers l'étude détaillée de la *Lexikon-Sonate*, c'est en combinant et associant des *Strukturegeneratoren* qu'Essl réalise de tels programmes.

La spécificité de la composition en temps réel réside dans cette rencontre à la fois de la composition algorithmique et du temps réel, et représente une rupture qualitative par rapport à l'une et à l'autre. En effet, l'utilisation d'algorithmes n'était auparavant possible qu'en amont du moment de réalisation de la pièce. Quant aux technologies en temps réel, celles qui existaient déjà en musique électronique ne pouvaient être

utilisées qu'en tant qu'instruments, pour produire et transformer du son. Les œuvres créées avec les prototypes d'ordinateurs en temps réel s'inscrivaient dans cette continuité, en utilisant la puissance de calcul pour élaborer de nouvelles transformations du son (par exemple *Répons* de Pierre Boulez ou *Antara* de George Benjamin). Essl a été l'un des premiers à mettre ces machines à calculer au service d'algorithmes de composition, et sans doute celui qui a le plus approfondi cette notion. Par la suite, avec la démocratisation de l'informatique (tant matérielle que logicielle), les pratiques similaires se sont multipliées, avec par exemple l'apparition au début du XXI^{e} siècle du *live coding* (utilisation d'environnements de programmation interactifs pour « improviser » des codes, écrivant ainsi devant une audience le code informatique générant la pièce entendue en même temps).

3.4 Descente de la tour d'ivoire

Le dernier événement décisif dans le parcours d'Essl a lieu en 1997, lorsqu'il est invité au festival de Salzbourg (*Salzburger Festspiele*) et présenté parmi les « compositeurs de la nouvelle génération ». Le festival de Salzbourg, dédié à l'opéra, au théâtre et à la musique classique, est un événement de premier plan. Il est de plus davantage tourné vers la musique historique (en particulier celle de Mozart, né à Salzbourg) que vers la création contemporaine ; l'invitation fête à Essl marque donc un sommet de sa notoriété de compositeur.

Cet aboutissement est cependant un coup d'arrêt pour Essl. Il décide alors de changer radicalement sa posture de compositeur, en quittant la « tour d'ivoire » dans laquelle il travaillait jusqu'alors. Au lieu de seulement composer seul des morceaux (ou des programmes) et d'en confier l'exécution à des interprètes (ou des machines), il souhaite s'ouvrir. Cela le conduit d'une part à devenir *compositeur-performeur*, d'autre part à chercher à collaborer avec d'autres artistes pour ses créations futures.

Pour être à nouveau confronté au public, Essl devient donc « *performeur* ». La performance est une pratique artistique présente dans toutes les disciplines de l'art contemporain, supposant des actions éphémères à caractère unique[28], dont par exemple

28. Au sens large, la musique générée par les programmes de composition en temps réel aurait donc un aspect performatif, au même titre que tout art de l'improvisation ; ce goût pour l'éphémère fait sans

John Cage ou Yoko Ono font partie des initiateurs. La désignation est cependant exagérée : les « performances » d'Essl sont avant tout des improvisations. L'improvisation fait partie des terrains de jeu privilégiés de la création musicale contemporaine, comme en témoigne par exemple le compositeur américain George Lewis : « [L'improvisation] se porte bien un peu partout. Beaucoup de monde essaie d'apprendre à improviser, parce que cela a toujours été un moyen formidable pour découvrir comment faire de la musique. Il y a une ébullition très intéressante parmi les improvisateurs — de nouvelles manières de structurer, des idées élaborées pour intégrer des partitions à l'intérieur de l'improvisation, de nouveaux sons, un élargissement des notions d'*instrument*, de *virtuose*, et du rôle du performeur. » [29].

La volonté d'Essl de jouer devant un public comme il l'avait fait avec les groupes de rock de sa jeunesse se heurte toutefois à un problème majeur : il n'a aucun instrument dont il pourrait jouer. Bien qu'il ait comme nous l'avons vu joué du piano, de la guitare électrique et de la contrebasse, il n'a en 1997 plus aucune pratique instrumentale de haut niveau. Là encore, il trouve la solution en capitalisant sur son savoir-faire informatique : « Il serait difficile d'imaginer mon travail sans les ordinateurs, même lorsque je fais de la musique instrumentale ; c'est grâce à l'ordinateur que j'ai pu quitter ma tour d'ivoire » [30]. Essl conçoit ainsi son propre « instrument », qu'il nomme *m@ze°2* (*Modular Algorithmic Zound Environment*). Programmer son propre instrument est une pratique commune dans la musique contemporaine, ainsi témoigne le compositeur américain Curtis Roads : « Beaucoup d'instruments digitaux peuvent être associés aux micro-ordinateurs de faible coût. La prolifération d'ordinateurs bon marché donne accès aux instruments intelligents à virtuellement n'importe quel musicien qui les souhaite. » [31].

Comme on peut le voir sur la photo 4, *m@ze°2* combine à la fois du matériel de musique électronique (pédales, micros, console MIDI. . .) et une interface logicielle (visible sur la photo 5) qui incorpore des algorithmes de traitement du son et des *Strukturgeneratoren* pour la composition en temps réel. Les deux aspects, traitement du son

---

doute partie des motivations d'Essl à se tourner vers cette pratique et à la désigner ainsi.

29. #Citation(George Lewis in Composers and the computer, p. 81)#
30. #Citation(Elektronische Musik / Komposition / Improvisation - Karlheinz Essl im Gespräch mit Silvia Pagano)#
31. #Citation(Composers and the computer, p. xvii)#

FIGURE 4 – Karlheinz Essl réalisant une performance avec *m@ze°2*— Photo © Viktor Brázdil

et composition algorithmiques, sont présents simultanément. Là encore, Essl réalise une synthèse dans une opposition classique décrite par exemple dans !!! : « [Tristan] Perich me dit que ses compositions émergent de l'improvisation, !!! — en général le piano, qui est son instrument de prédilection. Il oppose ceci à d'autres compositeurs qui utilisent des algorithmes, ce qui entraîne des complications. « Il y a une différence entre un processus qui fait partie de l'inspiration ou de votre ensemble d'outils, et un processus qui fait office de déterminant. » !!! Il préfère le premier. »[32] Le *m@ze°2* est donc un méta-instrument dans le sens employé ci-dessus. On peut même parler de « super-instrument ». Essl l'a en effet constamment développé et amélioré depuis sa création en 1998. On voit aussi sur la photo 4 qu'il s'en sert pour augmenter[33] des instruments traditionnels. Par rapport aux autres méta-instruments, *m@ze°2* ajoute de plus au seul programme informatique une véritable interface physique, tactile, qui permet à Essl de mettre au point une sorte de *geste* instrumental.

L'ouverture voulue par Essl passe aussi par la collaboration avec d'autres artistes, musiciens (instrumentistes et compositeurs) ou non. Cela l'amène à composer de nouvelles formes et pour de nouveaux cadres, comme l'écrit Julieanne Klein : « La production musicale d'Essl couvre tous les media possibles : orchestre, musique de chambre, théâtre musical, performance, musique électronique *live*, musique informatique et élec-

32. #Citation(Composers and the computer, p. 264)#

33. On parle en général d'*instrument augmenté* lorsqu'un musicien ajoute des fonctionnalités électroniques à son propre instrument — la guitare électrique est un instrument augmenté.

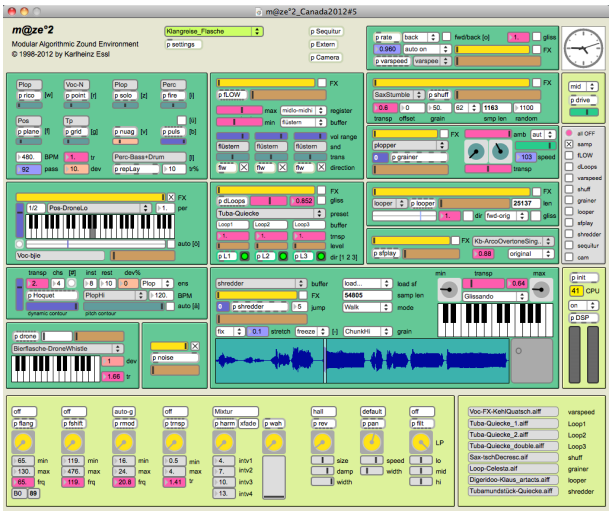

FIGURE 5 – L'interface graphique du logiciel *m@ze°2*— Photo © Karlheinz Essl

tronique, méta-compositions et compositions en temps réel, méta-instruments, installations et paysages sonores, musique de film, visuels, compositions textuelles et pièces pour instruments solo. Cherchant constamment à étendre sa production créative, Essl collabore fréquemment avec des artistes d'autres disciplines, y compris des chorégraphes, des danceurs, des plasticiens, des vidéastes, des architectes, des poètes, des écrivains, et des graffeurs. ». [34]

La concrétisation de cette nouvelle volonté d'ouverture passe par un projet d'envergure intitulé *fLOW*. Il s'agit d'une entreprise polymorphe autour d'un programme homonyme générant un paysage sonore ; elle donne lieu à dix installations et vingt-deux performances (principalement en 1999 et 2000, mais les dernières ont lieu en 2007). Toutes incorporent le *m@ze°2*, et chaque performance fait appel à des improvisateurs différents, jouant des instruments variés et issus de traditions musicales diverses (Nouvelle Musique, jazz expérimentale, improvisation livre, musique électronique).

Enfin, un autre événement important se produit en 1999 : l'ouverture du Essl Museum. Elle apporte deux explications supplémentaires à la nouvelle orientation de la carrière d'Essl. D'abord, elle s'inscrit dans la même ligne idéologique : Karlheinz Essl senior a fait construire ce musée pour mettre sa collection personnelle d'art contemporain à la disposition du public [35]. Sur le plan matériel, elle offre une plus grande autonomie à

34. #Citation(Julieanne Klein - A Portrait of the Composer Karlheinz Essl)#

35. La motivation de ce projet n'est peut-être pas seulement philanthropique — sa vocation pourrait par exemple aussi être une entreprise de communication —; elle n'est du moins certainement pas

Karlheinz Essl junior. Celui-ci dispose en effet d'un studio à l'intérieur du bâtiment, et d'un poste de conservateur chargé de la programmation musicale, grâce auquel il peut mettre ses productions en scène à l'intérieur des expositions, et nouer des relations en organisant des concerts.

3.5 Conclusion

=> CLASSIQUE DE SA GÉNÉRATION

#Citation(An Extended Composer's Desk - Composer Karlheinz Essl as the music curator of the Essl Museum)#

4 La composition en temps réel à travers ses programmes

4.1 Introduction

Cette partie se propose d'analyser l'œuvre d'Essl à partir du code même de ses programmes. étudier la notion de « composition en temps réel » dans l'œuvre d'Essl, ! ! ! ! ! !

4.2 La *Realtime Composition Library*

La *Realtime Composition Library* (*RTC-lib*) est une bibliothèque, c'est-à-dire un ensemble de fonctions réutilisables dans des programmes, développée par Karlheinz Essl dans le langage MAX/MSP. Son développement commence dès 1992, c'est-à-dire dès le moment où Essl a été mis en contact avec ce langage développé un an plus tôt par Miller Puckette (! ! !), lors de son stage à l'IRCAM. Son origine remonte même à 1988, si l'on considère qu'Essl a commencé par implémenter dans ce nouveau langage les algorithmes de composition qu'il avait déjà développés en xLOGO, rassemblés dans une bibliothèque intitulée *COMPOSE*.

économique, la construction du musée par l'architecte Heinz Tesar et son entretien étant assurés par le patrimoine de la famille Essl, sans aide publique. La mise à mal de ce patrimoine par la crise financière de 2007 à même conduit à un changement d'actionnaire majoritaire et à la décision de fermer le musée.

Cette bibliothèque est le socle de la plupart des œuvres réalisées par Essl impliquant la programmation. Il en recense 56 à ce jour : partitions, programmes, installations *works-in-progress*..., parmi lesquels certaines de ses créations majeures comme les *Sequitur*, *fLOW* ou encore la *Lexikon-Sonate*, étudiée dans la partie **??**. C'est aussi à partir d'elle qu'il a conçu l'« instrument » qu'il utilise lors de ses performances, le *m@ze°2*. Elle est distribuée sous licence libre : disponible en téléchargement, elle peut être utilisée et modifiée gratuitement, à la seule condition d'en citer les auteurs.

Dans ce qui suit, je propose une description détaillée et une analyse de cette bibliothèque, en commençant par ! ! !

4.2.1 Description

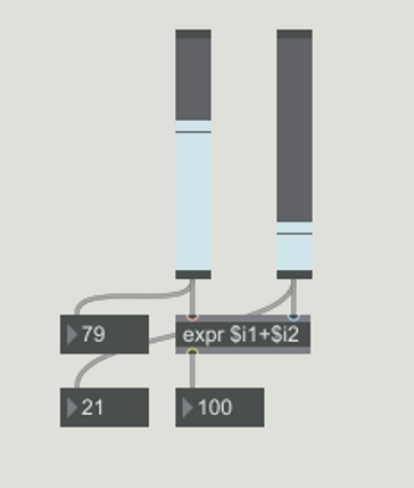

FIGURE 6 – Un programme basique en MAX/MSP : l'affichage et l'addition de deux entiers déterminés par des *sliders*.

MAX/MSP est un *langage de programmation graphique*, c'est-à-dire un langage de programmation dans lequel les programmes ne sont pas écrits en texte mais construits à partir d'éléments graphiques. Dans le cas de MAX, ces éléments sont des boîtes ou « *patches* » possédant des entrées et des sorties, et des liens qui permettent à ses fonctions d'interagir — typiquement : relier une sortie d'un *patch* à l'entrée d'un autre. La figure 4.2.1 montre un exemple de programme simple. Ce type de représentation des programmes comme un assemblage de *patches* reliés entre eux est la plus commune parmi les langages de programmation graphique, et on la retrouve dans d'autres environnements de CAO comme *OpenMusic*. Il existe deux modes d'édition distincts,

l'un pour construire le « circuit » du programme, l'autre permettant de l'exécuter et de modifier les paramètres (valeurs numériques, impulsions ou « *bangs* », curseurs etc.). MAX ayant été développé expressément pour la musique assistée par ordinateur, il contient de nombreux objets dédiés comme des entrées et sorties MIDI, ou dédiés au traitement du son [36].

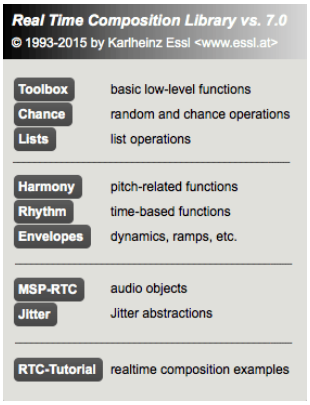

FIGURE 7 – Le menu et les huit rubriques de la *RTC-lib*. Photo © Karlheinz Essl

La *RTC-lib* compte à ce jour 177 objets en tout. 140 sont écrits par Essl, les autres en collaboration avec ou d'après d'autres compositeurs ou chercheurs [37]. Elle est divisée en huit rubriques, qu'illustre en outre un tutoriel (voir figure 4.2.1). Les trois premières rubriques, *Toolbox* (outils divers), *Chance* (opération aléatoires) et *Lists* (traitement de listes), contiennent des objets « basiques ». Les trois suivantes, *Harmony* (gestion de hauteurs de notes), *Rhythm* (gestion du rythme) et *Envelopes* (gestion du volume), des objets « de composition », et les deux dernières, *MSP* et *Jitter*, regroupent les fonctions directement liées aux entrées et sorties son et vidéo. L'analyse qui suit propose un découpage différent. Les fonctions des rubriques *Toolbox*, *Lists*, *MSP* et *Jitter* seront appelées *fonctions ergonomiques*. Elles répondent en effet au besoin de simplifier

36. En réalité, MAX ne permettait à l'origine que la manipulation de données au format MIDI ; c'est la bibliothèque MSP, ajoutée en 1997 afin de pouvoir l'utiliser avec la Station d'informatique musicale de l'IRCAM, qui permet le traitement de signal audio (*Digital Signal Processing, ou DSP*). Une autre bibliothèque d'importance nommée Jitter a été ajoutée en 2002, qui permet synthèse la manipulation graphiques.

37. R. Albert Falesch, Charles Baker, Frank Barknecht, John Chowing, Chris Dobrian, Richard Dudas, Gerhard Eckel, Peter Elsea, Philippe Gruchet, Gary Lee Nelson, Serge Lemouton, James McCartney, Iain Mott, Eric Singer, Les Stuck et David Zicarelli.

la tâche du programmeur en rendant plus rapides [38] ou plus intuitives des procédures d'usage courant, qui pourraient aussi servir à la réalisation de programmes en-dehors de l'informatique musicale. Les autres rubriques regroupent les *fonctions compositionnelles*, c'est-à-dire les fonctions qui résolvent des problèmes spécifiquement artistiques et musicaux. Aux trois rubriques qu'Essl fait entrer dans cette catégorie s'ajoute donc la rubrique *Chance*.

Les fonctions ergonomiques

La plupart de ces fonctions sont très simples, en particulier dans la rubrique *Toolbox*. On y trouve ainsi des fonctions réalisant les calculs de l'inverse d'un nombre, de son opposé, un accumulateur (fonction qui fournit la somme de toutes les valeurs qu'elle a reçues au cours de l'exécution du programme, ou le nombre de fois qu'elle a été appelée)... D'autres corrigent ou améliorent des fonctions pré-existantes, comme la division euclidienne (qui ne fonctionne à l'origine que si le numérateur positif) ou l'arrondi (rendu paramétrable par Essl : au lieu de calculer seulement l'entier le plus proche de la valeur d'entrée, on peut arrondir à x près).

C'est aussi le cas de la rubrique *Lists*, dont la plupart des fonctions proposent simplement une forme plus ergonomique de fonctions pré-existantes. 14 d'entre elles se résument même à un renommage de fonctions contenues dans un objet nommé *zl*. Celui-ci fournit de nombreuses procédures de manipulation de listes, mais aux noms peu explicites. *Dans la RTC-lib*, `zl rev` devient ainsi `reverse`, `zl rot`, `rotate`, ou encore `zl lookup` est légèrement modifiée pour devenir `nth`, qui renvoie le n-ième élément d'une liste fournie en entrée. La rubrique *Lists* contient ainsi principalement des opérations simples et classiques sur les listes : calcul de la longueur d'une liste, de l'intersection de deux listes, scission d'une liste... On en trouve d'autres qui résument en une seule fonction des opérations simples mais utiles et donc amenées à être utilisées souvent. Par exemple la fonction `butfirst` (respectivement `butlast`) cache simplement, en lui donnant une signification explicite, la scission d'une liste après son premier (respectivement avant son dernier) élément.

Enfin, les dernières fonctions essentiellement ergonomiques que contient la *RTC-lib* sont des conversions, présentes dans chaque rubrique. Elles permettent au composi-

38. Rapidité au moment de la création du code, pas de son exécution.

teur de manipuler les différents outils de la bibliothèque sans se soucier de leur compatibilité de type, qui peut être immédiatement gérée par ces convertisseurs. Cela va de transformations de *pitch* (hauteur de note) en note au format MIDI ou en fréquence à celles entre différentes sorties sons, ou encore à celles entre durées relatives et absolues.

Les fonctions compositionnelles

J'inclus la rubrique *Chance* dans cette catégorie, parce qu'à mon sens c'est dans celle-ci que réside le cœur de la partie véritablement compositionnelle de la *RTC-lib*. En effet, les fonctions que j'appelle *ergonomiques*, bien qu'elles aient leurs spécificités et reflètent des partis-pris (comme l'importance accordée à la structure de liste), sont en effet généralistes et répondent à des besoins d'abord *techniques*. La rubrique *Chance*, dédiée à la gestion de l'aléatoire [39], rassemble comme elles des outils basiques et des fonctions simples utilisables dans d'autres contextes ; mais les utilisations possibles du hasard sont si nombreuses que les choix faits dans cette rubrique ont nécessairement une portée esthétique, sur laquelle s'appuient les modules *Harmony*, *Rhythm* et *Envelopes*.

Plus précisément, elle rassemble principalement des générateurs aléatoires dont la variété rend possible une grande liberté dans le développement d'algorithmes musicaux. Là encore, l'ergonomie est une préoccupation centrale. Tous les générateurs sont ainsi déclinés en deux fonctions, selon qu'ils fournissent leurs résultats séquentiellement, ou directement sous la forme d'une liste. Là encore, toutes les fonctions ont un code relativement simple, parfois largement fondé sur des fonctions standard de MAX, comme la génération sérielle, présente nativement dans la fonction `xrandom` et rebaptisée `series`.

Les générateurs aléatoires présents dans la *RTC-lib* implémentent des lois de probabilités classiques. On retrouve ainsi la probabilité uniforme (chaque événement possible a la même probabilité, comme sur un dé équilibré), sérialisme (la probabilité qu'un événement qui a déjà eu lieu se reproduise est nulle jusqu'à ce que tous les autres événements se soient produits chacun une fois), chaînes de Markov (déplacement sur un graphe donné, ou les probabilités de passage d'un état à un autre sont fixées à

39. Cette notion est discutée dans la partie ? ? ?

l'avance), mouvement brownien (aussi appelé « marche aléatoire », déplacement au hasard dans un espace donné tel que deux positions successives soient « proches »). Certains de ces générateurs sont déclinés en plusieurs variantes intégrant des paramètres supplémentaires comme des pondérations ou des interdictions de répétition. On remarque aussi la présence moins commune de plusieurs générateurs reposant sur des échelles logarithmiques, ainsi que l'absence de générateurs reposant sur des distributions de probabilité comme les lois de Gauss ou de Poisson. D'une manière générale, on trouve presque exclusivement — c'est-à-dire à l'exception du générateur de mouvement brownien, *brownian* — des générateurs *discrets* et non *continus*[40].

Dans les rubriques qu'EssI qualifie lui-même de compositionnelles, on peut distinguer trois sortes de fonctions : celles qui implémentent directement des méthodes classiques de composition, celles qui adaptent les générateurs aléatoires à cette fin, et les « super-fonctions ».

Les premières fonctions sont ainsi spécifiques à la composition et au son, elles implémentent des comportements, des outils, des gestes classiques. Par exemple, la fonction `neutral-harmony`(n, i) génère séquentiellement des notes à partir de n par déplacement d'un intervalle i puis de son complémentaire ; la fonction `metro-dev%` modélise un instrumentiste humain en introduisant des approximations dans un battement parfaitement régulier ; la fonction `panning` calcule la distribution stéréophonique des volumes pour simuler le déplacement d'une source sonore. Parmi cette première sorte de fonctions, une sous-section entière de *Harmony* (16 fonctions) est dédiée aux manipulations dodécaphoniques usuelles (génération de séries, calcul de l'inverse ou de la rétrograde d'une série, etc.).

La deuxième sorte de fonctions est une application des différents générateurs aléatoires (et de certaines opérations de listes) à chaque rubrique. Par exemple, la fonction `brown-melody` génère séquentiellement des notes dont l'enchaînement suit un mouvement brownien, sans répétition d'octaves. Sur le plan purement technique, ce sont à peu de choses près des convertisseurs qui transforment les résultats de ces générateurs en données représentant des hauteurs de notes (entiers représentant des

40. Des générateurs dont les productions appartiennent à des ensembles prédéterminés et explicitement finis, par exemple les entiers entre 1 et 12, et non (théoriquement, l'ordinateur ne pouvant en réalité manipuler qu'un nombre très grand mais fini de données) infinis, par exemple l'ensemble des nombres réels $\mathbb{R}$.

pitches ou *pitch-classes*[41]), des rythmes (émission de *bangs*, ou entiers correspondant à des *entry delays* (ED)[42]), et des volumes (entiers entre 0 et 127). Néanmoins cette description est un peu réductrice, en ce que ces fonctions considérées de l'extérieur dépassent la simple conversion de données, possédant un véritable contenu sémantique.

Enfin, les « super-fonctions » combinent les précédentes. Elles n'apportent pas de possibilités créatives supplémentaires mais une interface pour choisir et alterner facilement entre les différents générateurs à disposition, en regroupant de plus de nombreux paramètres en entrée. C'est typiquement le cas du très complet `super-rythm`, visible sur la figure 4.2.1.

4.2.2 Analyse

L'analyse que je propose porte sur les aspects techniques et esthétiques de la *RTC-lib*. L'analyse technique étudie les choix directement liés à la programmation et leurs conséquences, ainsi que le « style » de conception de code. L'analyse esthétique montre le projet et les parti-pris artistiques que révèlent et servent ces choix techniques.

Analyse technique

Comme nous l'avons vu, la plupart des algorithmes présents dans la bibliothèque sont plutôt simples, mais il faut préciser ce que « simple » signifie. La manière théorique de quantifier la complexité d'un programme informatique est d'évaluer sa *complexité algorithmique*, qui correspond à l'ordre de grandeur du nombre maximal d'opérations élémentaires réalisées par une exécution en fonction de la taille de l'entrée[43]. Dans le

41. Le terme *pitch* renvoie à une hauteur de note au sens large, dans tout l'intervalle des fréquences audibles ; une *pitch-class* est l'une des douze hauteurs de la gamme. Un *pitch* correspond ainsi à l'association d'une *pitch-class* et d'un *registre*, qui indique dans quelle octave celle-ci se situe

42. L'*entry delay* d'un événement est le moment de son exécution dans la pièce, ou plus généralement la durée entre un point de départ global (comme le début du morceau ou d'une section) et le début de cette exécution.

43. Pour plus de détails sur cette notion centrale de l'informatique théorique, se reporter à **knuth1998art**

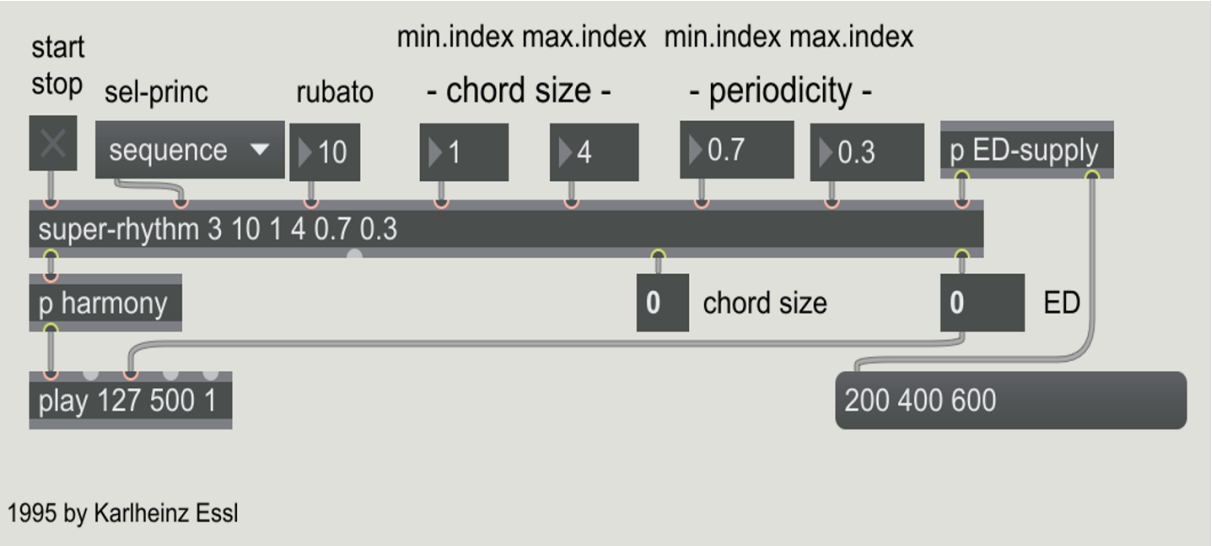

FIGURE 8 – La « super-fonction » `super-rhythm`, qui génère un rythme de manière largement paramétrable. Les ED générés sont sélectionnés parmi une liste fournie en entrée (*ED-supply*), selon un principe au choix (*sel-princ*, permet de sélectionner différents générateurs aléatoires). Une déviation aléatoire de ces valeurs d'origine peut être opérée (mesure par *rubato*). Enfin il est possible de régler la taille des accords (*chord size*, ou nombre de notes pouvant être jouées au même moment, donc au même ED) et la « périodicité » (*periodicity*), qui correspond à l'espérance du nombre de répétitions d'un ED immédiatement après avoir été sélectionné une première fois.

cas de la *RTC-lib*, tous les algorithmes ont une complexité *constante* ou *linéaire*[44], ce qui est effectivement faible. À titre de comparaison, de nombreux algorithmes courants comme le calcul du plus court chemin entre deux points d'un graphe ont une complexité algorithmique strictement supérieure. Les algorithmes de complexité constante s'exécutent en un nombre constant d'étapes, quelle que soit la taille de l'entrée (ou alors la taille des entrées est bornée). Les algorithmes de complexité linéaire nécessitent un nombre d'opérations élémentaires du même ordre de grandeur que la taille de l'entrée. En outre, les portions complexes (linéaires, donc) des fonctions de la *RTC-lib* ne sont jamais apparentes dans les *patches* établis par Essl ; si c'était le cas, elles apparaîtraient sous forme de boucles ou de récursivité (une fonction récursive peut s'appeler elle-même avant de terminer son calcul), totalement absentes : les calculs

44. Exception faite de la fonction `sort`, réécriture de la fonction `zl sort`, qui trie une liste. Le tri est un problème algorithmique extrêmement classique, et les algorithmes qui le réalisent ont généralement une complexité de l'ordre de grandeur de $(n \cdot \log n)$ pour une trier une liste de longueur n.

sont effectués de manière directe, sans retour en arrière[45]. Ainsi les seules sources de complexité sont cachées dans les sous-programmes (des fonctions natives) utilisés à l'intérieur des fonctions de la *RTC-lib*, principalement les manipulations de listes et les objets de type `metronome`.

La pensée algorithmique que révèle la *RTC-lib* est avant tout fonctionnelle. Cela signifie que le paradigme avec lequel Essl approche la programmation lui fait voir les algorithmes comme des fonctions qui *calculent* (pensée fonctionnelle) plutôt que par exemple comme des listes d'instructions à exécuter (pensée impérative). La fonction `remove` de la rubrique *Lists* illustre cette orientation : pour supprimer le n-ième élément d'une liste L, Essl calcule les sous-listes $L1$ et $L2$ séparées au niveau de cet élément, puis la liste $L1'$ comme $L1$ privée de son dernier élément, et enfin le résultat comme la jonction de $L1'$ et $L2$. Une pensée impérative aurait plutôt fait reculer d'un rang chaque élément à partir du $n+1$-ième. Que la pensée soit fonctionnelle n'a rien de surprenant car c'est la manière de programmer naturellement induite par le langage MAX, et associée avec l'utilisation de listes, omniprésentes dans la *RTC-lib*[46]. Il est cependant intéressant de noter qu'Essl, qui a commencé à programmer avec des langages impératifs (Basic, LOGO[47]) maîtrise parfaitement les codes du paradigme de programmation le mieux adapté dans ce contexte — malgré l'absence de récursivité, autre caractéristique essentielle de la pensée fonctionnelle.

Par ailleurs, les algorithmes sont généralement codés de manière directe, concise, et très claire. On peut même remarquer parfois le choix, augmentant cette concision, d'étapes *hard-coded* : au lieu de faire calculer une sous-fonction de manière procédurale, l'ensemble de ses valeurs est intégralement décrit en donnée[48]. Enfin, la do-

45. Les seules traces de récursion apparaissent lorsqu'il faut interdire certaines valeurs en sortie d'une fonction (répétition, octaves). Dans ce cas un élément fait office de « filtre », teste si le résultat est valide et, s'il ne l'est pas, lance une nouvelle exécution du reste de la fonction jusqu'à pouvoir finir une valeur autorisée. Ceci ne change pas significativement la complexité des fonctions concernées car la probabilité que les filtres doivent demander une nouvelle exécution est faible

46. La structure de données équivalente, en programmation impérative, étant le tableau

47. LOGO permet la création de fonctions, implémente la récursivité, utilise des listes, et repose sur un langage fonctionnel, le LISP ; sa philosophie générale est cependant plutôt impérative.

48. Comme si par exemple une fonction calculant le carré de son entrée x, au lieu de calculer $x \times x$, retournait la valeur de la x-ième case d'une liste fournie au départ de la forme $1, 4, 9, 16, 25, \cdots$). Dans la *RTC-lib* ce *hard-coding* est bien sûr utilisé pour simplifier des problèmes plus élaborés que cet exemple,

cumentation est très bien réalisée, avec des explications complètes mais claires de chaque fonction, l'organisation en rubrique, et la présence d'exemples ainsi que d'un tutoriel. Cette documentation, nécessaire à l'entretien du code à son appropriation par d'autre utilisateurs, tient à la volonté d'Essl de mettre effectivement cette bibliothèque à l'usage de tous.

L'analyse de la *RTC-lib* montre ainsi qu'Essl possède une bonne maîtrise technique de son outil, à la fois dans l'efficacité des algorithmes et dans la manière de les présenter et documenter.

Analyse esthétique

Il ne s'agit pas ici de faire une analyse esthétique approfondie, au sens courant, de la *RTC-lib* en tant qu'œuvre d'art contemporain, mais bien de mettre au jour les parti-pris de l'artiste Essl sous-jacents aux choix d'implémentation du programmeur Essl.

La première influence très nette est celle de l'école sérielle. Elle transparaît naturellement dans les fonctions directement dédiées aux calculs sériels (générateur aléatoire sériel, fonctions dodécaphoniques). Mais elle se ressent aussi à travers l'omniprésence des listes, structure de donnée naturellement adaptée à la représentation de séries, qui structurent l'ensemble des moyens de stockage de la bibliothèque. On peut en outre relever l'influence directe de Gottfried Michael Koenig et de Karlheinz Stockhausen ; plusieurs algorithmes font des emprunts à leurs travaux et ils sont mentionnés dans les commentaires.

Par ailleurs, la représentation structurelle de la musique est très « pianistique ». Les notes sont en effet décrites par uniquement une hauteur, une durée et un volume, ce qui modélise bien un piano mais pas par exemple un violon, dont le son résultant d'un seul geste musical peut évoluer dans le temps. Il aurait pu en être autrement compte tenu de l'intérêt qu'Essl accorde au son en général, mais cela n'a rien d'étonnant dans un langage de programmation qui privilégie la notation MIDI, et un langage musical marqué par la pensée sérielle.

Rythmiquement, on peut noter — ce qui est courant dans la musique du XX^{e} siècle — un affranchissement de la pulsation et de la notation sur partition, car ce ne sont ja-

par exemple pour calculer la distribution panoramique des volumes en fonction de la position supposée de la source sonore.

mais des noires, croches, doubles, etc., qui sont manipulées, mais exclusivement des durées absolues. Enfin, lorsqu'il s'agit de générer les ensemble de valeurs parmi lesquelles opérer des choix aléatoires, Essl a une préférence nette pour les échelles logarithmiques, disponibles presque systématiquement. Ce choix traduit vraisemblablement la volonté de permettre à la fois précision et contrastes.

En conclusion, le projet esthétique qui s'appuie sur la *RTC-lib* peut être qualifié de combinatoire. La bibliothèque est conçue comme une *boîte à outils* plus que comme une collection d'algorithmes élaborés. Les fonctions qu'elle contient sont directes, et faciles d'utilisation ; elles sont techniquement simples mais très complètes, offrant une grande liberté de composition. La composition d'une œuvre avec la *RTC-lib* passe ainsi par la combinaison de ces fonctions (à l'instar des modules de la *Lexikon-Sonate*, étudiés dans la partie **??**), et par le paramétrage des entrées qui peut engendrer des résultats musicaux très variés. Essl résume ainsi cette approche : « La plupart des objest [de la *RTC-lib*] sont orientés vers un traitement direct des données. L'utilisation de ces objets spécialisés rassemblés dans un *patch* clarifie et simplifie la programmation. Beaucoup de fonctions qui servent souvent en composition algorithmique sont disponible dans cette bibliothèque — ainsi le compositeur peut se concentrer sur la composition et non sur les aspects de programmation. » [49].

4.2.3 Aspects matériels du « hasard » et du « temps réel »

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Le « hasard »

#Citation(Philippe Codognet, New computational paradigms for computer music, p. 160)# => Ça se rejoint dans RTC ! ! ! LE HASARD N'EST PAS IMPORTANT MAIS LE RESSENTI DU HASARD => ART ET PAS SCIENCE RÉGRESSION PAR RAPPORT AU VRAI HASARD (DÉ) ÉLÉMENT DEVENU BANAL DU LANGAGE MUSICAL

Le « temps réel »

diese Ircam Signal Processing Workstation entwickelt. Das war ein NeXT Compu-

49. #Citation(RTC-lib)#

Année	**Mips** (! ! !)	**Modèle**	**Type** (! ! !)
1954	0,000640	IBM 704	Mainframes
1969	3,3	IBM 360/85	Mainframes
1973	0,065500	DEC PDP 11/45	Mini
1977	0,230	Apple II	Micro
1982	**2,188**	**Atari ST**	**Mini**
1985	1,6	VAX 11/785	Mini
1998	**525**	**iMac G3**	**Micro**
2003	3100	iMac G5	Micro
2008	50000	Apple Mac Pro	Micro

FIGURE 9 – Tableau MIPS ! ! !

ter mit einer speziellen Soundkarte, mit einem ganz tollen Prozessor, der konnte in Echtzeit Klangsynthese und Klangmanipulationen, Soundprocessing machen

#Citation(RTC-lib)#

#Citation(La musique du temps réel, p. 41)#

#Citation(La musique du temps réel, p. 42)#

#Citation(La musique du temps réel, p. 44-45)#

#Citation(Improvisation über "Improvisation" - Karlheinz Essl & Jack Hauser)#

4.3 La *Lexikon-Sonate*

La *Lexikon-Sonate* (commencée en 1992 et continuée jusqu'en 2007) est la pièce la plus connue de Karlheinz Essl. Elle naît de la rencontre de deux projets. D'une part, Essl vient de découvrir MAX et d'importer les algorithmes qu'il avait précédemment développés dans ce qui deviendra la *RTC-lib*. La *Lexikon-Sonate* lui permettra d'expérimenter l'application en temps réel de ces algorithmes dans une pièce complète. D'autre part, un groupe d'artistes viennois nommé *Libraries Of The Mind* conçoit un CD-ROM qui doit servir de support à un roman de l'auteur autrichien Andreas Okopenko (1930-2010), le *Lexikon-Roman*[50]. Cet ouvrage présente la particularité de nar-

50. **okopenko1983lexikon**

rer un récit fictif sous la forme d'un dictionnaire au travers duquel le lecteur est invité à voyager. L'idée du groupe *Libraries Of The Mind* est qu'il se prête particulièrement bien au format électronique, en tirant profit des liens hypertextes pour simplifier la lecture et les sauts d'une entrée à une autre. En outre, le projet propose d'enrichir le texte original en lui adjoignant des images et de la musique (faisant donc de ce CD un dispositif *multimédia*). Essl est donc invité à composer de courts extraits ou paysages sonores, chaque entrée du dictionnaire devant être accompagnée par un échantillon. Il refuse néanmoins cette approche, estimant que la musique accompagnant le *Lexikon-Roman* doit comme lui accorder une grande liberté au lecteur, par exemple changer nettement si celui-ci navigue rapidement d'une page à l'autre, ou au contraire conserver une ambiance stable si la lecture est calme.

C'est donc pour relever un double défi, technique et artistique, qu'Essl « compose », ou plutôt programme la *Lexikon-Sonate*. Dans ce qui suit, nous allons décrire la pièce et les différentes utilisations qui en ont été faites, puis proposer comme pour la *RTC-lib* une analyse d'une partie des algorithmes qu'elle emploie, et enfin dégager les conséquences que la forme de de la *Lexikon-Sonate* entraînent pour l'œuvre et sa réception.

4.3.1 Description

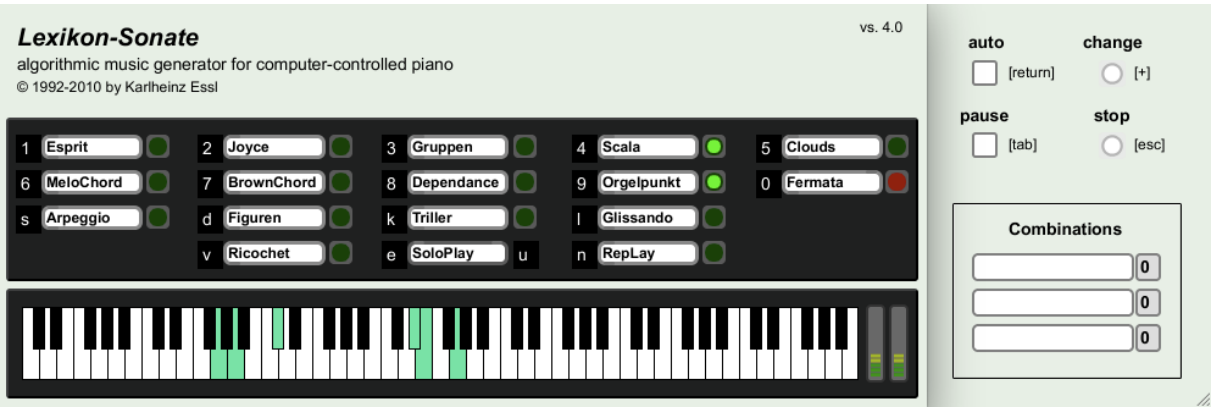

FIGURE 10 – Interface graphique de la *Lexikon-Sonate*. Photo © Karlheinz Essl.

La *Lexikon-Sonate* n'est pas un morceau au sens traditionnel du terme, mais un programme informatique qui génère de la musique. Reposant sur la *RTC-lib*, elle utilise

en temps réel des algorithmes de composition algorithmique pour générer de la musique au format MIDI, qui est alors ou bien directement convertie en son de piano de synthèse par l'ordinateur sur lequel le programme est exécuté, ou bien jouée par un piano mécanisé comme le *Yamaha Disklavier*. Du fait de l'utilisation du hasard à plus ou moins grande échelle par les algorithmes implémentés en MAX par Essl, cette musique est n'est ni prédéterminée ni prévisible, et se renouvelle donc sans cesse. Le programme génère donc à chaque exécution un morceau qui pourrait être sans fin, et se crée au fur et à mesure qu'il est joué sans jamais se répéter.

Comme on peut le voir sur la figure 4.3.1, la *Lexikon-Sonate* utilise 17 « modules » [51] (intitulés *!!!*, *!!!*, etc.) équivalents. Chaque module est un algorithme, construit en combinant des fonctions de la *RTC-lib*, qui génère. Ce sont ce qu'Essl nomme des *Strukturgeneratoren* : le produit de chaque module respecte une forme, une structure préétablies. Les dites structures recouvrent une large variété de fonctions et de complexités musicales. GLISSANDI, qui génère des *glissandi*, c'est-à-dire une série ascendante ou descendant de notes « consécutives » (séparées d'un ton ou d'un demi-ton)« Il s'agit de la définition de la structure de *glissando* telle qu'elle apparaît dans la structure de ce module, et non de la définition générale, plus large et recouvrant des réalités différentes selon les instruments. », est relativement simple à décrire, tandis qu'ESPRIT vise à une mélodie *expressive* au sens de la musique classique viennoise. Les différentes structures peuvent selon les modules s'apparenter à des mélodies, des accompagnements, des « textures » ou des « figures » ; certains comme FERMATA sont même dédiés au silence, permettant à la musique composée de reprendre son souffle et de la segmenter, faisant entendre l'équivalent de parties et de sous-parties.

À chaque instant, modules exactement sont actifs. Ils génèrent chacun une séquence musicale correspondant à leur structure, de manière indépendante, qui sont jouées simultanément. Un changement des modules sélectionnés peut ensuite s'opérer (voir ci-dessous). La pièce jouée est ainsi la résultante du produit de trois *Strukturgeneratoren* totalement autonomes. Pour conférer une unité auditive à cette superposition, qu'elle ne soit justement pas entendue comme une confrontation absurde de sons

51. Il en existe en réalité 24 en tout. Mais la figure montre la version libre de la *Lexikon-Sonate* ; les 24 modules ne sont disponibles que dans une version plus complète qu'Essl réserve pour ses propres performances et celles d'artistes qui lui en font la demande.

mais paraisse véritablement téléologique dans sa construction, les trois modules actifs reçoivent par ailleurs une pondération (« premier plan », « second plan », « arrière-plan »), qui influence les paramètres réglant la gestion aléatoire du volume et de la proportion de silence. Essl affirme que les structures définies par les modules présentent des similarités avec des langages musicaux de l'histoire du piano, des *topoï* de cet instrument, « de Johann Sebastian Bach, Mozart, Beethoven, Schönberg, Webern, Boulez, [et] Stockhausen [à] Cecil Taylor ». Bien qu'il n'existe aucune citation directe des pièces pour clavier historiques, le programme est conçu de manière à ce que des formes musicales proches de celles-ci apparaissent au gré du hasard.

Il existe différentes manières de faire jouer le programme, autrement dit différents *modes de jeu*. À l'origine, la *Lexikon-Sonate* avait été imaginée comme une œuvre du type installation, avec une interactivité très limitée. La seule intervention possible de l'utilisateur était de provoquer le changement des modules actifs, qui s'opérait par remplacement du module présent depuis le plus longtemps, en suivant un principe sériel (lorsqu'un module qui est activé, il ne sera plus réactivé tant que tous les autres modules ne l'ont pas été). C'est ce principe qui, associé avec le *Lexikon-Roman*, permet de régler l'homogénéité de la musique sur le rythme de lecture. Un autre mode de jeu sans interactivité est implémenté, dans lequel les moments de remplacement de modules sont déterminés de manière aléatoire. Plus tard, Essl élargit le champ des possibles du programme et en fait un méta-instrument, en permettant à l'utilisateur de choisir lui-même quel module est activé et à quel moment. Cette version est distribuée librement en ligne ; dans une version plus élaborée réservée aux performances, le programme peut être relié à un contrôleur MIDI grâce auquel des paramètres globaux comme la « densité » du son (proportion statistique de moments de silence) peuvent être réglés par l'interprète-performeur.

Le choix du piano pour cette pièce n'est pas anodin. Essl parle d'une revanche envers un instrument qu'il n'avait pas aimé apprendre lorsqu'il était enfant [52] ; c'est d'ailleurs un

52. « L'un des points de départ est ma relation entre amour et haine avec cet instrument, le piano. À sept ans, j'ai été obligé d'apprendre le piano ; je voulais jouer de la flûte à bec. Je ne suis cependant jamais devenu bon pianiste. C'était toujours très frustrant de voir que j'étais incapable de mettre adéquatement en œuvre sur les touches ce que j'avais dans la tête. Bien sûr, mes premières recherches de compositions furent cependant des pièces pour piano, mais depuis je n'ai écrit aucun autre morceau pour le piano, et j'envisage difficilement d'en écrire à nouveau. Tout ce que je pouvais faire pour le piano,

instrument pour lequel il a relativement peu écrit [53]. On peut dégager trois motivations à ce choix. D'abord, le projet d'associer la pièce au *Lexikon-Roman* invite à adopter une démarche artistique proche de celle qu'il emploie. Or celui-ci est imaginé comme un pied de nez à la littérature classique, puisque le *Lexikon-Roman*, c'est-à-dire le roman-dictionnaire n'est pas un roman mais bien un dictionnaire, qui contient cependant des références aux formes classiques. Le choix du piano permet d'évoquer l'histoire et le répertoire extrêmement riches de cet instrument ; le titre, *Lexikon-Sonate*, ou sonate-dictionnaire, fait elle aussi allusion à la forme extrêmement classique de la sonate mais n'a évidemment aucun rapport avec elle dans sa construction. Elle ressemble de fait plus à un dictionnaire, avec cette banque de modules, qui permettent en outre à Essl de mettre l'idée de *Strukturgenerator* à l'épreuve au travers de cette recherche de *topoï* historiques. La deuxième raison est la justification du principe de superposition de trois modules simultanés. Comme le piano, ou plus généralement tout clavier, est l'instrument polyphonique traditionnel par excellence, sur lequel par exemple on « réduit » les œuvres orchestrales lorsqu'un instrumentiste seul fait en faire entendre la teneur musicale, il est plus aisé d'écouter la rencontre de structures potentiellement contradictoires lorsqu'elle s'opère avec le timbre de cet instrument. Enfin, le piano est aussi l'instrument dont les modèles automatisés comme le *Yamahe Disklavier* sont le plus répandus. Il s'agit par conséquent d'un choix naturel dans la conception d'une œuvre sous forme de programme contrôlant un instrument.

4.3.2 Étude de deux modules : `Triller` et `MeloChords`

Cette partie propose, comme ci-dessus pour la *Realtime Composition library*, une étude détaillée du programme de la *Lexikon-Sonate*. Plus précisément, c'est TRILLER, l'un des « modules » ou *Strukturgeneratoren* qui composent la pièce, qui fait l'objet d'une analyse complète. Un autre module intitulé MELOCHORD est ensuite décrit de manière moins approfondie mais de façon à dégager les points communs avec TRILLER.

c'était une pièce qui obéisse à des critères absurdes. » #Citation(Der Wiener Komponist Karlheinz Essl (Hanno Ehrler))#

53. Outre la *Lexikon-Sonate*, on recense seulement cinq pièces pour piano : *Con una certa espressione parlante* (1985), *Take the C Train* (2009), *juncTions* (2011/2012), *Gold.Berg.Werk* (2010-2014) et *STERN* (2013). Et *Take the C Train* et la seule autre œuvre pour piano solo (dans la mesure où l'on peut désigner ainsi la *Lexikon-Sonate*).

`Triller`

Le module TRILLER est un générateur de *trilles*. Traditionnellement, un trille est un ornement musical réalisé en alternant rapidement deux notes consécutives (séparées d'un ton ou d'un demi-ton). Cette définition est ici élargie : il s'agit d'un alternance rapide des notes d'un ensemble de 2 à 7 éléments, couvrant au plus une octave.

La figure 4.3.2 montre les deux parties principales du générateur de trilles [54]. Il faut d'abord déterminer aléatoirement les différents paramètres du trille (quelles notes, à quel volume, etc.), et ensuite celui-ci peut être généré en temps réel.

Comme on le voit sur la première partie de la figure 4.3.2, la génération est commandée en premier lieu par le rythme. La fonction [`p rythm`] émet un signal à chaque moment où une note doit être jouée, qui active les fonctions [`p notes`], [`p dynamic`] et [`p duration`], qui déterminent chacun un paramètre (respectivement la hauteur, le volume et la durée) de cette note. Le synthétiseur [`QT-SndPlayer`] reçoit ces informations et exécute la note correspondante. La subordination au rythme des autres paramètres du son dans l'ordre de détermination a d'importances conséquences sur le processus de composition. Il est en effet plus compliqué ainsi de construire une ligne mélodique et de lui attribuer ensuite un rythme, par exemple de ralentir à la fin [55]. Cette remarque montre que, bien que permettant en théorie de créer toutes sortes d'algorithmes (ne seraient-ce que tous ceux réalisables par un compositeur humain, sans ordinateur), les environnements de CAO sont des outils comme les autres, dont les contraintes orientent les parti-pris de leur utilisateur.

En tout, sept paramètres régissent le trille produit, visibles en bas du deuxième cir-

54. Les figures sont tirées des fonctions programmées par Essl en MAX/MSP. Précisons pour le lecteur qui souhaiterait les comprendre plus en détail que certaines liaisons entre sont masquées pour plus de visibilités. En outre, les *patches* dont le nom commence par la lettre `s` (*send*) ou `r` (*receive*) font de même office de liaison : la valeur reçue par [`s label`] est transmise à [`r label`]. Enfin, les noms de *patches* commençant pas `p` sont locaux : ceux-ci masquent un circuit complexe, mais le même nom utilisé à l'intérieur d'une fonction différente ne correspondra pas au même circuit. Par exemple, les deux *patches* nommés [`p rhythm`] dans les deux images de la figure 4.3.2 correspondent à des circuits différents. Ils sont seulement nommés ainsi pour indiquer que les circuits qu'ils représentent traitent tous deux d'aspects rythmiques.

55. Cela est plus compliqué ou moins évident, mais reste possible. Il faudrait placer le générateur mélodique en amont, qui fournisse son produit à la fois à une liste et au générateur de rythme. Les signaux émis par celui-ci commanderaient alors l'envoi progressif des notes de la liste au synthétiseur.

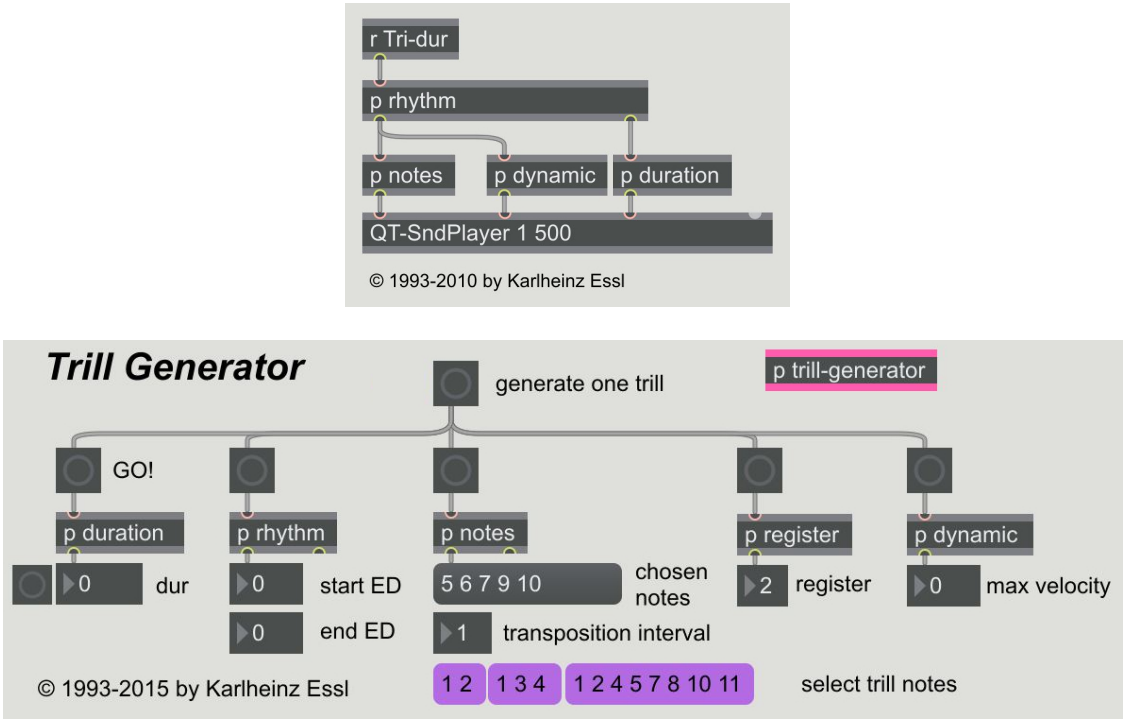

FIGURE 11 – Les deux parties principales du programme du module TRILLER. L'image du haut correspond à la génération des notes du trille en temps réel ; celle du bas regroupe les paramètres qui commandent celle-ci. Le circuit du haut est masqué par le *patch* rose [`p trill-generator`] de celui du bas.

cuit de la figure 4.3.2 : *dur*, *start ED*, *end ED*, etc. Ici, ils sont déterminés lorsque le bouton central est activé, et dans la *Lexikon-Sonate*, à chaque fois que le module est appelé. La durée totale du trille (*dur*), les durées minimale et maximale entre deux notes successives (*start ED* et *end ED*, décidées par la fonction visible à gauche de la figure 4.3.2), le volume maximum (*max velocity*), l'intervalle de transposition (*transposition interval*) et le registre (*register*) suivent une loi de probabilité uniforme parmi l'ensemble de leurs valeurs possibles ; c'est la fonction [`between`] de la *RTC-lib* qui opère la sélection. Pour les trois premières, comme on peut le voir sur la figure 4.3.2, les valeurs possibles sont réparties sur une échelle logarithmique entre les deux valeurs extrêmes, calculée par la fonction [`trans-log`]. La détermination du matériel harmonique est plus élaborée, présentée à droite de la même figure. L'ensemble des notes de la gamme chromatique, représentée par les entiers de ! ! ! à ! ! !, est d'abord mélangé aléatoirement par la fonction [`scramble`]. La fonction [`between`] sélectionne uniformément le nombre de notes à conserver entre 2 et ! ! !, et le transmet à la fonction [`slice`] qui scinde adéquatement la liste mélangée. Le résultat est enfin trié ([`sort`]). Cet algo-

rithme permet effectivement de construire un ensemble aléatoire avec une probabilité uniforme (tous les produits possibles ont une probabilité identique). D'autres manières de parvenir au même résultat, y compris avec des fonctions de la *RTC-lib*, étaient possibles, par exemple en choisissant des notes parmi les douze possibles selon le principe sériel. La méthode employée par Essl témoigne à nouveau d'une pensée fonctionnelle plutôt qu'impérative.

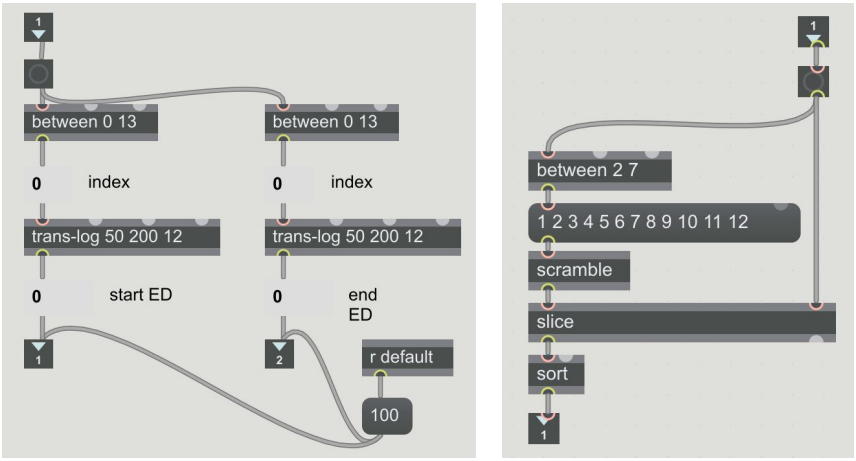

FIGURE 12 – !!!

Ces paramètres sont fixés pour tout le trille ; c'est à partir d'eux qu'est calculée chaque note. Certains des dits calculs utilisent des générateurs aléatoires. Le hasard intervient ainsi à deux échelles différentes dans le programme. Le rythme suit, à nouveau, une échelle logarithmique, ce dont il résulte un effet global de ralentissement. Cette échelle est calculée par la fonction [`ED-trans`] de la *RTC-lib*, afin que les durées entre chaque note varient de *start ED* à *end ED*, sur une durée totale de *dur* millisecondes. À chaque signal rythmique, une note est générée. L'une des hauteurs pré-sélectionnées est choisie au hasard avec interdiction de répétition par la fonction [`permutate`]. Elle est ensuite « additionnée » aux deux paramètres *transposition interval* et *register* qui définissent la hauteur absolue de la note la plus basse du trille, et permettent donc à celui-ci d'être exécuté sur potentiellement n'importe quelle partie du clavier. Le calcul du volume sur l'ensemble du trille utilise la seule fonction [`schweller`], qui réalise un *crescendo* suivi d'un *decrescendo* entre le silence complet et le paramètre *max velocity*. La durée de chaque note est aussi établie de manière déterministe, comme une fois et demie la durée qui la sépare de la suivante, ce qui donne un effet « lié » au résultat entendu.

`MeloChords`

Le module MELOCHORD génère des mélodies dont les notes peuvent éventuellement être enrichies par un accord. La figure 4.3.2 montre que le principe d'organisation est identique à celui rencontré dans TRILLER. Un premier circuit génère les paramètres globaux de la mélodie, et un second crée celle-ci note par note, sous l'impulsion d'un générateur rythmique.

Ce module est visiblement plus complexe que le précédent, régi par douze paramètres différents. Il s'agit cependant d'une complexité *horizontale*. Chaque génération de note implique plus de valeurs en entrée mais la profondeur du circuit est du même ordre de grandeur, et dans les deux cas il n'y a pas de boucle, comme dans la *RTC-lib*.

À nouveau, les paramètres généraux sont majoritairement sélectionnés à partir de distributions uniformes, souvent parmi des échelles logarithmiques. Les hauteurs de notes utilisent des intervalles calculés par l'une des fonctions purement musicales de la *RTC-lib*, [`choose intervals`], qui garantit des propriétés sonores du matériau généré (la somme de deux intervalles ne peut être une octave, on ne peut pas construire d'accord diminué d'après ces intervalles, etc.). Pour les paramètres de distances entre les notes (ED), de registre, et de durée (de chaque note cette fois, et non plus de la structure totale), on remarque la présence d'une option nommée « périodicité ». Elle correspond à la probabilité pour chaque nouvelle note de conserver les valeurs de la précédente plutôt que de faire un nouvel appel au générateur aléatoire, afin d'obtenir par moments des unités sonores. Sur la figure 4.3.2, la « périodicité » est élevée (0.91) pour les notes courtes et nulle pour les notes longues. Ainsi, lorsqu'une note courte est créée, il est probable qu'elle commence une phrase enchaînant des notes de même durée. D'une manière générale, le produit de ce module devant être plus varié que celui de TRILLER, il fait appel à plus de fonctions complexes de la *RTC-lib*. La génération du volume fait ainsi appel à un mouvement brownien. Quant au générateur de rythme, il utilise la fonction [`repchord-rhythm`], qui dépend de six paramètres : les valeurs minimales et maximales d'ED, le nombre de pas pour l'échelle logarithmique calculée entre ces extrêmes, deux valeurs de « périodicité » qui varie sur cette échelle, et la taille maximale des accords générés (autrement dit le nombre de notes qui peuvent être générées simultanément).

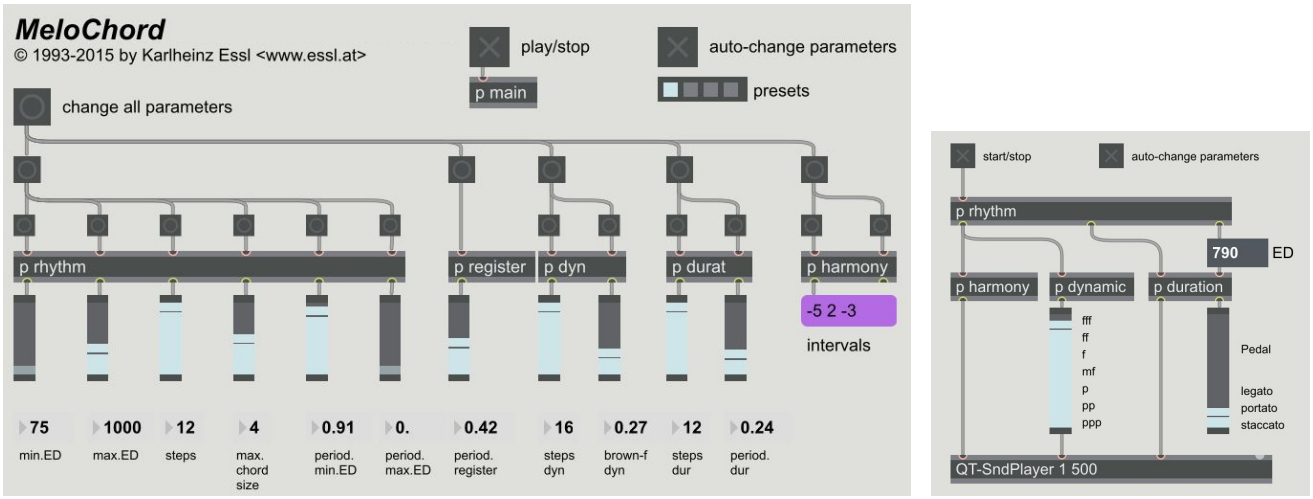

FIGURE 13 – Les deux parties principales du programme du module MELOCHORD. L'image de droite correspond à la génération des notes en temps réel ; celle de gauche regroupe les paramètres qui commandent celle-ci. Le circuit de droite est masqué par le *patch* [`p main`] de celui de gauche.

Conclusion

DIRECT COMME LA RTC-LIB

UTILISE BIEN LA RTC-LIB

HASARD SUR DEUX NIVEAUX

4.3.3 Analyse

À nouveau, l'objectif de cette partie n'est pas de proposer une analyse esthétique approfondie de la pièce, mais de déterminer les spécificités d'une telle œuvre, en particulier dues à l'utilisation inédite de l'outil informatique.

ENTRETIEN –> WELTANSCHAUUNG

La première caractéristique évidente de la *Lexikon-Sonate* est qu'il s'agit d'une *œuvre ouverte*. Cette notion assez large qualifie une œuvre dont certaines facettes ne sont volontairement pas maîtrisées ou arrêtées par l'auteur, mais laissées à celui qui la reçoit ou à des agents extérieurs — en particulier, il s'agit généralement d'œuvres qui abandonnent l'idée d'un « message » à transmettre au profit d'un champ d'interprétations possibles. La pièce *4'33"* de John Cage est ainsi un exemple minimal et éloquent d'œuvre ouverte : elle consiste en quatre minutes et trente-trois secondes de la part de

l'interprète, tout en affirmant chaque son produit dans la salle de concert pendant l'exécution de la pièce fait partie de celle-ci ; l'auteur est alors entièrement dépossédé de son œuvre qu'il soumet à la contingence des conditions de sa réalisation. Le concept est développé par Umberto Eco dans *L'Œuvre ouverte*. Eco y fait remarquer que la définition peut être appliquée à n'importe quelle œuvre d'art, puisque la représentation mentale qu'elle induit chez celui qui la consomme varie d'un individu à l'autre, étant nécessairement soumise à l'interprétation. Mais nombre d'œuvres du XX^{e} siècle se caractérisent par la mise en exergue de cette aspect intrinsèque de l'œuvre d'art.

Pour montrer que la *Lexikon-Sonate* s'inscrit effectivement dans une problématique d'œuvre ouverte, nous allons mettre en avant trois caractéristiques essentielles d'une œuvre ouverte, que cette pièce possède effectivement : l'indétermination, la mise en retrait de l'auteur, et la remise en question de la notion même d'œuvre d'art.

L'indétermination, la place laissée à la contingence, est essentielle dans l'idée d'œuvre ouverte. Elle s'oppose en cela aux œuvres « fermées » produites par les siècles précédents, c'est-à-dire les œuvres univoques, que critique par exemple Wittgenstein : « Il me vient à l'esprit que dans les discours sur les objets esthétiques, des expressions telles que sont employées : "Il faut voir cela ainsi, c'est pensé ainsi", [...] "Tu dois entendre cette mesure comme une introduction", [etc.] (cela vaut aussi bien à propos de l'écoute que du jeu). » [56]. Eco décrit la réponse suivante des artistes à cette critique : « Aujourd'hui, l'accent est mis sur le processus, sur la possibilité de saisir *plusieurs ordres*. Dans la réception d'un message structuré de façon ouverte, l'*attente* implique moins une *prévision de l'attendu* qu'une *attente de l'imprévu*. » [57]. Il est clair que l'omniprésence de l'élément aléatoire, rendue possible seulement par l'ordinateur et une pièce sous forme de programme informatique, crée cet imprévu en permanence.

La mise en retrait de l'auteur, de l'artiste, qui renonce en quelque sorte à sa position de démiurge pour partager la paternité de sa production (typiquement avec le récepteur de l'œuvre, soit par l'interactivité, soit par l'interprétation dans une perspective proche du constructivisme radical cher à Essl), est un autre aspect important de l'œuvre ouverte. On pourrait la confondre avec l'indétermination ou du moins penser qu'elle est impliquée par elle, mais il n'en est rien. Si Essl avait construit sa pièce comme un

56. #Citation(Zur Theorie der Seriellen Musik in Strukturgeneratoren - Algorithmische Komposition in Echtzeit)#

57. Eco, *Opera aperta : Forma e indeterminazione nelle poetiche contemparanee*, p. 105

unique *Strukturgenerator*, sa position de créateur aurait été plus affirmée : certes le hasard déterminerait certaines modalités, mais l'auteur serait en mesure de décrire la structure de n'importe quelle exécution. Dans la *Lexikon-Sonate*, le retrait est véritablement opéré par l'intrication des trois modules indépendants. À travers ce choix, Essl accepte l'impossibilité de prévoir même l'effet général produit par la pièce, et d'être lui-même surpris. Cette caractéristique, à la différence des deux autres étudiés ici, relève d'un parti-pris artistique et non de l'exploitation des nouvelles voies ouvertes par l'outil informatique.

Enfin, les propriétés de l'œuvre ouverte font qu'elle interroge la définition même de ce qu'est une œuvre. Adorno écrit ainsi : « Contrainte par la logique de ses propres faits, la musique, en un mouvement critique, a dissous l'idée d'œuvre achevée et rompu avec le public. [...] Les seules œuvres qui comptent aujourd'hui sont celles qui ne sont plus des « œuvres ». »[58]. Bien que ce ne soit pas si extrême dans le cas de la *Lexikon-Sonate*, le fait qu'un programme informatique soit présenté comme un morceau de musique, renforcé par la référence à la forme historique de la sonate, questionne effectivement les délimitations traditionnelles de l'art. Elle affirme — comme beaucoup d'autres créations contemporaines — la possibilité de nouvelles acceptions de l'œuvre qui enrichissent la tradition artistique plutôt que de nécessairement rompre avec elle.

Une autre spécificité de la *Lexikon-Sonate* découlant de sa forme informatique est l'absence de partition. L'enjeu de la notation a traversé toute l'histoire de la musique et trouve dans cette pièce un écho inédit. La partition classique, à l'origine très simple et conçue comme un simple aide-mémoire, a été sans cesse enrichie à travers les siècles, avec la volonté permanente d'une part de décrire de plus en plus précisément les paramètres d'interprétation, d'autre part de s'affranchir de ses contraintes (la notation du rythme sous formes de noires, croches, etc., est par exemple très contraignante). C'est pourquoi de nombreux compositeurs du XX^{e} siècle ont élaboré de nouveaux systèmes de notation, comme Stockhausen qui créait en créait un spécifiquement pour chaque nouvelle pièce. Renoncer à la partition présente l'avantage de pouvoir explorer la musique sous un nouvel angle (typiquement pouvoir manipuler d'autres hauteurs de son que les seules douze notes de la gamme chromatique), mais

58. Adorno, « Philosophie der neuen Musik », p. 41-42

revient aussi à renoncer à décrire l'œuvre ainsi créée d'une manière intelligible par n'importe quel musicien. Les conditions d'exécution sont alors beaucoup plus difficiles à réunir. Ce n'est pas le cas avec la *Lexikon-Sonate*, dont la notation est le programme lui-même, qui présente l'avantage d'être un système simple et dépourvu d'ambiguïté pour son « interprète » : l'ordinateur.

Le fait que la *Lexikon-Sonate* se passe d'interprète à d'autres conséquences. En particulier, la composition étant affranchie des contraintes de l'instrument, une nouvelle « virtuosité » est possible. Par exemple, un pianiste humain ne peut jouer de *glissando* que sur les seules touches blanches ou les seules touches noires. Dans la *Lexikon-Sonate*, le module GLISSANDI généralise complètement cette définition en générant n'importe quelle suite monotone de notes séparées d'au plus un ton. De même, la figure 4.3.3 montre un extrait de la musique générée par le programme telle qu'elle pourrait être reportée sur une partition traditionnelle. Dans les deux cas, résultat n'est pourtant pas complexe à l'audition ; simplement, la musique peut être déterminée grâce à des modèles qui n'ont plus à tenir compte des limitations physiques de l'instrument et de l'interprète. Cette liberté dans la composition n'est bien sûr pas spécifique à cette pièce ni nouvelle. Elle est permise par la machine mais ne nécessite pas le temps réel. La nouveauté que l'on trouve ici est à nouveau la synthèse opérée par la composition en temps réel : la composition algorithmique permet de prendre une partie de la responsabilité du compositeur, l'automatisation (de même que tous les procédés d'enregistrement) permet de se passer d'interprète, et dans l'exécution de la *Lexikon-Sonate*, tous deux sont absents. En outre, cela permet de rendre infini ce qui ne pouvait être auparavant que fini. Lorsque la composition nécessitait plus de temps que l'exécution, les pièces devaient être délimitées dans le temps. Ici, composition et exécution sont simultanées : c'est grâce à cela — et bien sûr au fait que l'exécution est confiée à une machine, qui « ne se fatigue jamais » [59] —que la musique peut se renouveler perpétuellement.

Enfin, cette redéfinition du schéma classique compositeur-interprète-auditeur a pour conséquence une nouvelle forme de sociabilisation [60] autour de l'œuvre d'art. D'abord,

59. « Un ordinateur ne se faitgue jamais. Avec l'ordinateur, on peut mettre en scène une musique qui ne s'arrête jamais. Cela n'est évidemment pas possible avec des musiciens. » #Citation(Elektronische Musik / Komposition / Improvisation - Karlheinz Essl im Gespräch mit Silvia Pagano)#

60. D'autres remarques sur les aspects sociaux du projet d'Essl, notamment à travers sa communica-

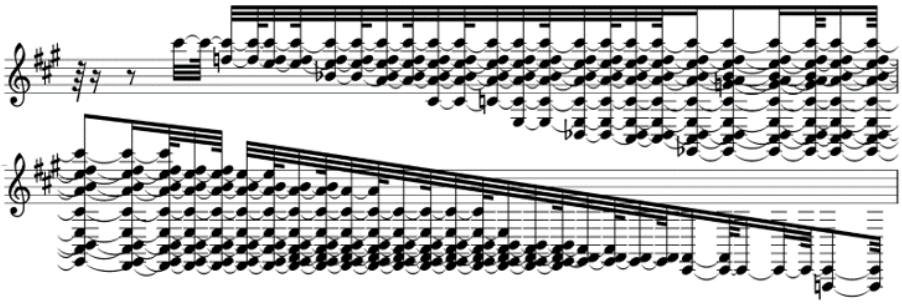

FIGURE 14 – Environ deux secondes de musique jouée par la *Lexikon-Sonate* transcrites sur une partition traditionnelle. Le son produit est simple (un arpège descendant avec les notes tenues), mais la partition paraît extrêmement compliquée. Cela est simplement dû à la liberté rythmique qui ne trouve pas d'équivalent dans ce système de notation.

grâce au caractère d'ouverture de la *Lexikon-Sonate*, la hiérarchie entre l'artiste est son audience est en partie effacée. Essl écrit ainsi : « Je veux mettre l'auditeur au défi de ne pas simplement consommer la pièce, mais tout en l'écoutant de devenir une sorte de co-créateur, un partenaire du compositeur et de la composition elle-même » [61]. Cette vision idéalise les faits : l'autorité du compositeur reste importante, mais l'auditeur est effectivement invité à approcher la pièce de manière plus active et ludique en tirant profit de l'interactivité (limitée) du programme. Cette interactivité oriente en outre vers une écoute « savante », ou du moins de réduire la possibilité d'une écoute véritablement naïve, puisque l'utilisation des appels de modules entraîne nécessairement une connaissance de base du fonctionnement de la pièce. Par ailleurs, l'absence d'interprète permet, comme la musique enregistrée, de diffuser plus facilement la *Lexikon-Sonate*, et donc de toucher un public potentiellement plus large, d'autant plus que la pièce est proposée en téléchargement gratuit. Mais les différents auditeurs-utilisateurs auront chacun une expérience inévitablement différente, ce qui fait qu'ils ont moins en commun que s'il s'agissait d'une musique fixée.

CONCLUSION ? ? ?

tion sur le réseau Internet, font l'objet de la section **??**.

61. #Citation(Profile Karlheinz Essl - Karlheinz Essl in conversation with Joanna King)#

4.4 Essl et la culture numérique

Cette dernière partie traite d'un aspect peu abordé jusqu'ici du rôle de l'ordinateur dans le travail d'Essl, l'utilisation qu'il fait du réseau Internet. Les enjeux en sont moins technologiques que sociaux. La façon dont le réseau Internet mondial, qui est l'un des piliers de l'informatique au sens large et le principal moteur de la culture numérique, bouleverse l'activité du compositeur, mérite cependant d'être étudiée. Internet ouvre de nouvelles perspective dans le rapport à l'œuvre, dans le lien avec le public et avec les autres artistes. Chez Essl, la redéfinition du rapport à l'œuvre passe avant tout par la question de l'auctorialité. Celui avec le public est PLAN ! ! !

TOPO HISTORIQUE => INTERNET C'EST LA MÊME PÉRIODE

La vitesse des communications et l'effacement des barrières géographiques qu'elle entraîne modifient radicalement les perspectives traditionnelles de circulation des informations. Ils font par exemple apparaître le problème de la surveillance et de la protection de la vie privée, ou celui du téléchargement illégal. Dans le cas d'Essl, comme de tout artiste, c'est justement la question du statut auctorial et de la distribution de ses œuvres — partitions, enregistrements, programmes — qui est posée par le monde numérique. Sa position est intéressante. D'un côté, il cède une grande partie des privilèges traditionnels de l'auteur. Ainsi distribue-t-il la plupart de ses programmes [62] sous licence libre, comme nous l'avons vu dans le cas de la *Lexikon-Sonate* et dans celui de la *RTC-lib*, qui sont pourtant respectivement son œuvre au plus grand succès et le socle logiciel de tout son travail algorithmique de composition. De même, Essl rend ses partitions disponibles et gratuites. Il met à disposition sur son site personnel de nombreux enregistrements de ses pièces, ainsi que sur sa chaîne YouTube où il rend publiques des performances, des entretiens et des cours (exactement 300 séquences vidéo à ce jour). Ce parti-pris s'expliquer grandement par le contexte historique. Comme cela a été mis en avant dans la partie 3.2, Essl fait partie de la génération qui a connu la généralisation des micro-ordinateurs et l'idéologie de liberté à laquelle elle s'associe. À la même époque, dans le courant des années 1980, apparaît en outre le courant du logiciel libre [63], qui promeut le partage des ressources logicielles de tous vers tous

62. Seuls le *REplay PLAYer* et *fLOW* sont des *sharewares*, c'est-à-dire qu'il en existe deux versions, l'une gratuite mais au fonctionnalités limitées, l'autre payante et complète.

63. Plus précisément, ce sont les logiciels propriétaires qui apparaissent, le logiciel étant auparavant

et gratuitement, auquel Essl s'associe donc dans la distribution de ses travaux que nous venons d'évoquer. Chacun de ses programmes téléchargeables inclut ainsi une licence délimitant précisément les usages que peut en faire l'utilisateur, ce qui montre la préoccupation du compositeur sur cette question. Toutefois, il affirme dans le même temps son statut auctorial ainsi que sa propriété intellectuelle. Il ne s'oppose pas à ce que certaines exécutions de ses œuvres soient payantes, touchant des droits sur les concerts ou CD dans lesquels d'autres musiciens jouent ses compositions. Il lui arrive aussi, comme nous l'avons vu dans le cas de la *Lexikon-Sonate*, de conserver une version améliorée de ses programmes exclusivement pour ses propres performances ; dans le cas de son instrument *m@ze°2*, il n'existe même pas de version publique. Un événement dans le développement de la *Lexikon-Sonate* est révélateur de l'intérêt porté par Essl à la question auctoriale : à l'origine, le programme possédait une sortie MIDI, mais Essl a découvert que guilldes personnes l'ont utilisé pour générer des partitions. Comme si c'était un programme générateur de partitions. Et ils ont ainsi créé des morceaux qu'ils ont vendu comme leurs propres morceaux [64]. Essl a donc désactivé la sortie MIDI de la version publique, consolidant ainsi sa position d'auteur.

L'une des spécificités du réseau Internet est qu'il « met en jeu un type de liaison multimodale entre les utilisateurs sans précédent : la liaison de *tous vers tous* [où] chaque internaute est à la fois récepteur et émetteur » [65]. Remarquons avant tout qu'Essl n'exploite cependant pas tout ce potentiel : s'il comme nous allons le voir il tire profit de la liaison de *un vers tous* pour la diffusion de son œuvre, il a peu recours à celle de *tous vers un*, c'est-à-dire les possibilités d'interactivité. En ce sens, son utilisation du réseau mondial est la même que celle des media de masse (radio, télévision, cinéma), dont les propriétés sont simplement amplifiées par Internet. Cette limitation de l'interactivité est d'ailleurs une caractéristique générale de l'œuvre d'Essl. Contrairement à beaucoup d'autres compositeurs, il n'en fait pas un enjeu esthétique important ; ses programmes et installations permettent une interaction mais l'utilisateur a des possibilités restreintes plutôt qu'une liberté totale. Même son instrument *m@ze°2* n'est pas interactif au sens large : bien que certaines de ses fonctionnalités aient recours au ha-

considéré comme un objet scientifique et circulait comme tel dans le milieu universitaire. Pour plus de détails sur l'histoire du logiciel libre, voir **paloque2013histoires**

64. #Citation(ENTRETIEN)#

65. Edmond Couchot et Norbert Hillaire, *L'art numérique* (Flammarion, 2003), p. 63

sard, celles-ci repose toujours sur l'idée de *Strukturgenerator*, et produisent donc une réponse imprévisible mais jamais imprévue aux stimuli qu'il reçoit.

On peut identifier dans la démarche d'Essl une véritable tentative de démocratisation de son travail, de le rendre accessible à la fois en termes intellectuels et logistiques. Dans sa démarche esthétique, ce pas vers le public passe par le refus d'une musique trop abstraite [66], l'utilisation de matériau musical connus de tous [67], ou d'une manière générale l'attachement au son et à l'effet qu'il produit à l'intérieur de la démarche algorithmique [68]. Sur le plan pratique, il s'incarne dans la recherche de lieux de concerts non-traditionnels comme les musées, en particulier le Essl Museum, et naturellement dans l'utilisation d'Internet pour diffuser son œuvre. Les aspects de « liberté » déjà évoqués dans la diffusion d'enregistrements et de programmes y participe. De plus, Essl multiplie les lieux virtuels de sa présence sur Internet : les réseaux sociaux Twitter et Facebook, les plate-formes de distribution audio et vidéo SoundCloud et YouTube, ainsi que sa page personnelle [69]. Celle-ci fait office à la fois de vitrine et de musée — « L'Internet apparaît [...] comme le lieu de [la] mémoire à venir, comme musée virtuel global. » [70]. Essl y propose, outre des liens pour télécharger ses programmes ou consulter ses séquences vidéos et enregistrements, sa propre actualité (compositions, concerts), de nombreux éléments biographiques, une liste exhaustive de ses écrits, et un recensement des publications à son propos. Cela représente un travail considérable qui lui permet de faire sa propre muséification. On peut cependant se demander

66. « Je ne fais pas de la musique abstraite qu'on ne peut comprendre que si l'on a une formation spéciale, ou des connaissances particulières, ou beaucoup d'expérience. Ce qui m'intéresse, c'est avant tout ce qu'il y a d'immédiat, d'extatique, qui se révèle à celui qui est prêt à s'abandonner. » #Citation(Rückblick / Vorschau - Der Komponist Karlheinz Essl im Gespräch mit Annelies Kühnelt)# #Citation(+ Karlheinz Essl / Bernhard Günther - Realtime Composition - Musik diesseits der Schrift)#

67. « Je voulais délibérément utiliser un matériau très succinct, qui soit de plus facile à identifier. [...] Il existe ainsi certaines musiques très caractéristiques, comme aussi la *Cinquième* de Beethoven ; [...] il suffit de les jouer et chacun sait immédiatement de quoi il s'agit. » #Citation(Wagner in Translation - Karlheinz Essl im Interview mit Annegret Huber)#

68. « Je crois que l'algorithme ne doit jamais devenir un fétiche se suffisant à lui même, à la logique duquel il faudrait faire appel pour dire : cette structure est si belle, si mathématique à tous points de vue, si cohérente. Cela doit toujours passer à l'épreuve de la réalité et de notre expérience d'auditeurs. »#Citation(Intuition, Automation, Entscheidung. Der Komponist im Prozess algorithmischer Komposition)#

69. http://www.essl.at, sur laquelle on trouvera quelques remarques dans

70. Couchot et Hillaire, *L'art numérique*, p. 231

si cette démarche n'est pas dans une condition nécessaire de visibilité donc d'existence dans un monde globalisé, saturé de communication et soumis à une très forte concurrence, y compris dans le domaine artistique. De même, on peut s'interroger sur la portée de la démocratisation qu'elle vise, car comme le ! ! ! : « L'ambitieuse volonté d'une communication universelle est contrecarrée par la réalité. Le public attendu, innombrable, multi et transculturel, reste encore un public de spécialistes très liés au monde de l'art et très identifiables socialement. Les "communautés virtuelles" sont en fait des microcosmes assez fermés, avec déjà leurs traditions et leurs orthodoxies. » [71].

INTERACTIVITÉ : NADA

COMMUNICATION + IL EST PROF IL TRANSMET

CONCLUSION IL Y A L'ŒUVRE MAIS ESSL NE S'EFFACE PAS => AI

#Citation(net.music)#

#Citation(NET Music - Karlheinz Essl talking to Golo Föllmer)#

#Citation(Julieanne Klein - A Portrait of the Composer Karlheinz Essl)#

#Citation(Elektronische Musik / Komposition / Improvisation - Karlheinz Essl im Gespräch mit Silvia Pagano)#

4.5 Conclusion : Essl et l'ordinateur

RAPPORT À L'ORDINATEUR : OUTIL (=> MAC)

C'EST UN COMPOSITEUR QUI A FAIT UN PROGRÈS ESTHÉTIQUE LE PROGRÈS TECHNOLOGIQUE C'EST LES ORDIS ET MAX

#Citation(Composing in a Changing Society - How does a composition come into existence ?)#

#Citation(Irreal-Enzyklopädie - Bernhard Günther - Einer metaphorischen Reise zur Lexikon-Sonate von Karlheinz Essl)#

#Citation(Intuition, Automation, Entscheidung. Der Komponist im Prozess algorithmischer Komposition)#

71. ibid., p.79

5 Conclusion

Résumé

CECI EST UN RÉSUMÉ

Mots-clefs : 1 2 3 4 5 6

6 Bibliographie

Références

Adorno, Theodor W. « Philosophie der neuen Musik » (1949).

Appleton, Jon, Curtis Roads et John Strawn. *Composers and the Computer*, 1986.

Assayag, Gérard, et Andrew Gerzso. *New computational paradigms for computer music*. T. 17. Delatour, 2009.

Breton, Philippe. *Histoire de l'informatique*. La découverte Paris, 1987.

Cauquelin, Anne. *Les théories de l'art : « Que sais-je ? » n. 3353*. Presses universitaires de France, 2010.

Collins, Nick. *Introduction to computer music*. John Wiley & Sons, 2010.

Collins, Nick, Alex McLean, Julian Rohrhuber et Adrian Ward. « Live coding in laptop performance ». *Organised sound* 8, n° 03 (2003) : 321–330.

Cope, David. *The algorithmic composer*. T. 16. AR Editions, Inc., 2000.

Couchot, Edmond, et Norbert Hillaire. *L'art numérique*. Flammarion, 2003.

Daston, Lorraine. *Things that talk : Object lessons from art and science*. MIT Press, 2004.

Daston, Lorraine, et Peter Galison. *Objectivity*. Zone books, 2010.

Dobretsberger, Christine. « Karlheinz Essl : Der Komponist als Zufallsgenerator ». In *Mozarts Erben*. Ibera Verlag, 2006.

Donin, Nicolas, Laurent Feneyrou et Pierre-Laurent Aimard. *Théories de la composition musicale au XXe siècle*. T. 1. Symétrie, 2013.

Eckel, Gerhard. « About the Installation of Karlheinz Essl's Lexikon-Sonate ». 1995. `http://www.essl.at/bibliogr/lexson-eckel.html`.

———. « Technological Musical Artifacts ». `http://iem.at/~eckel/publications/eckel98a/eckel98a.html`.

Eco, Umberto. *Opera aperta : Forma e indeterminazione nelle poetiche contemporanee*. T. 3. Tascabili Bompiani, 1962.

Essl, Karlheinz, et Annelies Kühnelt. « Rückblick / Vorschau ». *AKKORD (Zeitschrift der Musikschule Klosterneuburg)* 1 (2006/2007).

Felber, Andreas. « Der verlängerte Schreibtisch des Komponisten ». In *PASSION FOR ART, Ausstellungskatalog*. Sammlung Essl Privatstiftung, 2007.

Förster, Jonas. « Intuition, Automation und Entscheidung. Der Komponist im Prozess algorithmischer Komposition ». Mémoire de maîtrise, Folkwang Universität der Künste Essen, 2011.

Hiller, Lejaren Arthur, et Leonard M Isaacson. *Experimental Music ; Composition with an electronic computer*. Greenwood Publishing Group Inc., 1979.

Jamie, James. *La Musique des sphères*, 1997.

Karl, Popper. *La quête inachevée*, 1981.

Klein, Julieanne. « A Portrait of the Composer Karlheinz Essl ». *Fowl Feathered Review* 4 (2013) : 74–81.

Kuhn, Thomas S. *The structure of scientific revolutions*. University of Chicago press, 2012.

Lacoste, Jean. *La philosophie de l'art : « Que sais-je ? » n. 1887*. Presses universitaires de France, 2010.

Manoury, Philippe, Omer Corlaix et Jean-Guillaume Lebrun. *La musique du temps réel : entretiens avec Omer Corlaix et Jean-Guillaume Lebrun*. Editions MF, 2012.

Miller, Arthur I. *Colliding worlds : how cutting-edge science is redefining contemporary art*. WW Norton & Company, 2014.

Moles, Abraham. « Art et ordinateur ». *Communication and langages* 7, n° 1 (1970) : 24–33.

Puckette, Miller. « Combining event and signal processing in the MAX graphical programming environment ». *Computer music journal* 15, n° 3 (1991) : 68–77.

Schillinger, Joseph. *The Schillinger system of musical composition*. C. Fischer, inc, 1946.

Scholl, Steffen. « Karlheinz Essls RTC-lib ». In *Musik – Raum – Technik. Zur Entwicklung und Anwendung der graphischen Programmierumgebung »Max«*, 102–107. Transcript Verlag, 2014.

Sinkovicz, Wilhelm. « Fantasie als Sprengstoff ». *DIE PRESSE* (2005).

Weberberger, Doris. « Porträt : Karlheinz Essl ». *MUSIC AUSTRIA* (2012).

Wilson, Stephen. *Art+Science Now*. Thames & Hudson, 2012.

Wittgenstein, Ludwig. *Recherches philosophiques*. Editions Gallimard, 2014.

Wodon, Bernard. *Histoire de la musique*. Larousse, 2014.

Xenakis, Iannis. « Musiques Formelles Nouveaux Principes Formels de Composition Musicale » (1981).