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

31 mai 2016

Table des matières

1 Introduction

2 Enjeux et présentation

Enjeux

#Citation(Breton, avant-propos)#

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

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

Musique et informatique

2.1 Musique algorithmique

3 Portrait de Karlheinz Essl

3.1 Parcours

INTRO

3.1.1 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 ? ? ?)

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

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 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].

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 [3] 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. 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**[4], auquel l'a initié son professeur Friedrich Cerha[5]. 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 va de paire 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[6]. 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*[7] et du hasard, problématiques très présentes dans l'œuvre d'Essl. Koenig a même affirmé même que « la maîtrise du hasard par le com-

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

5. 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)#

6. #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)

7. 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 »[8]. 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 ? ? ?

? ? ? Est-ce qu'il faut mettre les dates de tous ces gens ?

3.1.2 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.

8. #Citation(.)#

C’EST QUOI LA MUSIQUE ALGORITHMIQUE ?

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

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

#Citation(OMNIA IN OMNIBUS : Behind the Scenes - Karlheinz Essl im Gespräch mit Katharina Hötzenecker)#

REMARQUE : EN MÊME TEMPS QUE TOUT LE MONDE

Théorique => pratique

3.1.3 IRCAM et Composition en temps réel

LEXIKON-SONATE

MAZE

#Citation(Composers and the computer, p. xvii)#

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

PRODUCTIVITÉ

3.1.4 Ouverture –> Internet

IMPROVISATION

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

#Citation(George Lewis in Composers and the computer, p. 81)#

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

3.1.5 Conclusion

=> CLASSIQUE DE SA GÉNÉRATION

3.2 Esthétique

3.2.1 INTERNET ? ? ?

3.2.2 Multitâche

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

3.2.3 Autariat ! ! !

#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*.

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*, que!!!. 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. Elle est disponible en téléchargement et il est possible de l'utiliser et de la modifier 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

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. Un exemple (figure!!!). 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 [9].

La RTC-lib est divisée en huit rubriques, qu'illustre en outre un tutoriel (voir figure!!!). Les trois premières rubriques, *Toolbox*, *Chance* et *Lists*, contiennent des objets « basiques ». Les trois suivantes, *Harmony*, *Rhythm* et *Envelopes*, des objets « de composition », et les deux dernières, *MSP* et *Jitter*, regroupent les fonctions directement liées aux sorties son et vidéo. À ce jour il y a en tout 177 objets. 140 sont écrits par Essl, les autres en collaboration avec ou d'après d'autres compositeurs ou chercheurs [10].

9. En réalité, MAX ne permettait à l'origine que la manipulation de données au format MIDI ; c'est la bibliothèque MSP, ajoutée en 1997, 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.

10. 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,

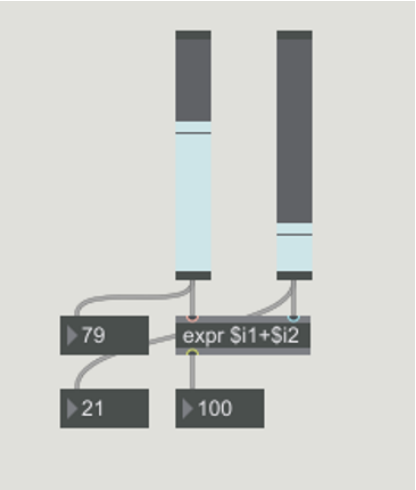

FIGURE 1 – Un programme basique en MAX/MSP : l'affichage et l'addition de deux entiers déterminés par des *sliders*.

On y trouve de nombreuses fonctions très simples, en particulier dans la rubrique *Toolbox* : calcul de l'inverse, de l'opposé, accumulateur... D'autres corrigent ou améliorent des fonctions pré-existantes, comme la division euclidienne (ne fonctionnant à l'origine qu'avec un numérateur positif) ou l'arrondi (rendu paramétrable par Essl).
C'est aussi le cas de la rubrique *Lists*, dont la plupart des fonctions proposent simplement une forme plus ergonomique de fonctions pré-existantes, parfois simplement en leur donnant un nom plus clair. Elle contient ainsi des fonctions calculant la longueur d'une liste, l'intersection de deux listes, scindant une liste... On en trouve d'autres qui résument en une seule fonction des opérations très 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 avant tout pratiques que contient la RTC-library sont des conversions présentes dans chaque rubrique. Elles permettent au compositeur 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 en note MIDI ou en fréquence à celles entre différentes sorties sons, ou encore à celles entre durées relatives et absolues.

Iain Mott, Eric Singer, Les Stuck et David Zicarelli.

4.2.2 Analyse

Théorique

Technique

Esthétique Le cœur de la partie véritablement compositionnelle de la RTC-library réside à mon sens dans la rubrique *Chance*. En effet, bien qu'elle contienne comme *Toolbox*, *Lists*, *MSP* et *Jitter* des outils basiques représentant des fonctions simples, c'est la variété des générateurs aléatoires présents dans celle-ci qui rend possible la liberté de développer des algorithmes musicaux. Et c'est pour beaucoup à partir de cette variété que se construisent les fonctions des modules *Harmony*, *Rhythm* et *Enveloppes*. Là encore, l'ergonomie est une préoccupation importante ; 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* rebaptisée *series*.
On retrouve la plupart des générateurs aléatoires classiques : probabilité uniforme, sérialisme, chaînes de Markov, mouvement brownien. Certains sont déclinés en plusieurs variantes intégrant des paramètres supplémentaires (pondérations, interdiction de répétitions...). On remarque la présence moins commune de plusieurs générateurs reposant sur des échelles logarithmiques, mais 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.

Si l'on se penche maintenant sur les rubriques dites « de composition », on peut y distinguer trois sortes de fonctions.
D'abord, il y a les fonctions spécifiques aux données du son concernées, qui implémentent des comportements et des outils 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, tandis que la *metro-dev%* modélise un instru-

mentiste humain en introduisant des approximations dans un battement parfaitement régulier, et que la fonction *panning* calcule la distribution stéréo des volumes pour simuler le déplacement de la source sonore. Parmi cette première sorte, toute une sous-section de *Harmony* est dédiée aux manipulations dodécaphoniques usuelles.
La deuxième sorte de fonctions est une application des différents générateurs aléatoires et de certaines opérations de listes à chaque rubrique. Concrètement, ce sont à peu de choses près des convertisseurs qui transforment les résultats de ces générateurs en données représentant des hauteurs de notes (entiers d'un certain ensemble), des rythmes (*bangs* émis, ou entiers correspondant à des ? ? ? ? ? ? ? ? ? ? ? ? ?ED), et des vélocités (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, possèdent un véritable contenu sémantique. Les fonctions de la troisième sorte regroupent les précédentes. Elles possèdent de nombreux paramètres d'entrée, dont la possibilité de choisir parmi les différentes méthodes aléatoires utilisables. C'est typiquement le cas du très complet *super-rythm*.

J'ai écrit que la plupart des algorithmes présents dans cette bibliothèque était plutôt simples ; en terme de complexité algorithmique, ils sont tous au plus linéaires (ce qui est, effectivement, une complexité faible). Cette complexité n'est en outre jamais apparente dans les algorithmes d'Essl (sous forme par exemple de boucles ou, plus évident en MAX, de récursivité), mais seulement dans les sous-procédures élémentaires qu'il appelle, plus précisément les manipulations de listes et les objets de type *métronome*.

S'il fallait décrire un « style » de programmation dans la RTC-library, je dirais que c'est une pensée 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). Une illustration simple en serait la fonction *remove* de la rubrique *Lists* : 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. Cela 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-library (la structure de données équivalente de la programmation impérative étant le tableau). Il est cependant intéressant de noter qu'Essl, qui a commencé à programmer avec des langages impératifs (Basic, LOGO) 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. D'une manière générale, lesl algorithmes sont 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 procédurielle, l'ensemble de ses valeurs est intégralement décrit en donnée.

Enfin, quelques remarques sur les choix esthétiques sous-jacents. L'influence de l'école sérielle est très présente, ainsi que celles de Gottfried Michael Koenig et de Karlheinz Stockhausen, à qui plusieurs algorithmes font des emprunts (ils sont nommés dans les descriptions). La pensée est aussi très « pianistique » dans le sens où les notes sont décrites par d'uniques hauteur et vélocité (comme sur un piano mais pas sur un violon, où le son d'une même note 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 un affranchissement de la notation sur partition car ce ne sont jamais 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 - sans doute parce qu'elles permettent à la fois précision et contrastes -, disponibles presque systématiquement.

En conclusion, il apparaît que les moyens techniques nécessaires à l'utilisation de cette bibliothèque en temps réel concernent avant tout l'interaction entre l'unité de calcul et les périphériques de son ou d'image. Les algorithmes utilisés par Essl sont d'une complexité très en-deçà non seulement des capacités non seulement des processeurs actuels, mais aussi de leurs prédécesseurs. [Il faudrait que je creuse pour avoir une idée de ce qu'on peut exécuter en temps perçu comme réel sur les *personal computers* des trente dernières années, mais je soupçonne que c'était déjà le cas,

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
2003	3100	Mac G5	Micro
2008	50000	Apple Mac Pro	Micro

FIGURE 2 – Tableau MIPS ! ! !

ou alors presque, en 1991.] Il possède une bonne maîtrise technique, à la fois dans l'efficacité des algorithmes et dans la manière de les présenter et documenter (sans quoi l'entretien du code et son utilisation par d'autres serait beaucoup moins aisée). Mais c'est son savoir-faire de compositeur qui ressort, qui se reflète dans le choix des algorithmes plutôt que dans leur implémentation. Bien que ceux-ci soient « simples » au sens informatique, la RTC-lib est très complète car d'une part chaque algorithme de composition peut produire des résultats très divers selon la leur paramétrage, et ils sont facilement combinables, pour créer des générateurs de musique dont les modules de la *Lexikon-Sonate* sont d'éloquentes illustrations.

4.3 Analyse

#Citation(RTC-lib)#

4.4 Quelle « composition en temps réel » ?

#Citation(Philippe Codognet, New computational paradigms for computer music, p. 160)# => Ça se rejoint dans RTC ! ! !

#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)#

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

#Citation(RTC-lib)#

4.5 La *Lexikon-Sonate*

#Citation(L'Œuvre ouverte, p. 30)#

#Citation(L'Œuvre ouverte, p. 105)#

#Citation(Adorno, p. 41-42)#

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

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

#Citation(Der Wiener Komponist Karlheinz Essl (Hanno Ehrler))#

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

4.6 Conclusion : Essl et l'ordinateur

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

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

5 Démocratisation ?

#Citation(Kuhn)#

#Citation(Les théories de l'art, p. 25)#

#Citation(Breton, intro chap 11)#

#Citation(Breton, p. 206)#

#Citation(L'art numérique, p. 63)#

#Citation(L'art numérique, p. 79)#

#Citation(L'art numérique, p. 231)#

#Citation(? ? ?)#

#Citation(net.music)#

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

#Citation(Technological Musical Artifacts (Gerhard Eckel))#

#Citation(Der Wiener Komponist Karlheinz Essl (Hanno Ehrler))#

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

#Citation(Wagner in Translation - Karlheinz Essl im Interview mit Annegret Huber)#

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

#Citation(Rückblick / Vorschau - Der Komponist Karlheinz Essl im Gespräch mit Annelies Kühnelt)#

#Citation(Karlheinz Essl / Bernhard Günther - Realtime Composition - Musik diesseits der Schrift)#

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

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

6 Conclusion

Résumé

CECI EST UN RÉSUMÉ

Mots-clefs : 1 2 3 4 5 6

7 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 contemparanee*. 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).