CLOS (Common Lisp Object System)による作曲支援及び音響合成の統合環境：IRCAM OpenMusicの今日

CLOS（Common Lisp Object System）による作曲支援及び音響合成の統合環境
～ IRCAM OpenMusic の今日～

土屋　雄

はじめに

CLOS（Common Lisp Object System）は記号処理による人工知能（AI）の技術として1980年代はじめ頃から注目されはじめ，教育，科学技術等の分野で科学者や数学者たちに広く利用されている，オブジェクト指向のプログラム言語である。音楽の分野では1980年代の半ばから欧米の大学や研究機関等でその環境構築が幾度も繰り返し試みられ，2000年に入ってからはその研究も次の段階に進められ現在もなお展開し続けている。その中でも有名なものとして，アメリカのスタンフォード大学で研究そして開発された Common Lisp Music，Common Music，Common Music Notation とフランスの IRCAM（Institute for music / acoustic research and coordination）の OpenMusic，PatchWork，FORMS，Modalys といった環境が挙げられるが，ここでは中でも殊に目覚ましい進化を続ける IRCAM の OpenMusic を中心に実際に作業しながらこれまでの伝統的な音楽，或いは音響に対して如何に共有し，更に拡張していくかを述べていく。

1．作曲（音楽）環境と CLOS

1-1．作曲支援とコンピュータ

コンピュータによる作曲支援というものは電子・コンピュータ音楽史上で比較的新しい分野である。ほとんどの研究機関においては音響分析，音響合成といった分野を中心に可能性を追求されてきた。（もちろん現在もこれらの分野が主とした研究対象となっていることは否めないが。）このことが考えられる要因は幾つかあるだろうが，その１つとして，おそらく作曲家とエンジニア或いはプログラマーとのコラボレーションの在り方，つまりその密接度の深さにあるのではないだろうか？

この作曲支援という分野は両者の対話なしでは決して成り立たない。言うまでもないことだが，現代の芸術音楽においては作曲家がソナタ，フーガといった構造，そして機能和声等のシステムの様な決まりきった秩序に沿って創造されることは稀である。このことは作曲家

の数だけその構造或いはそのアイディアの秩序，システム等が存在するといっていいだろう。その様な現状の中で技術的には過去の作品をモデル化することは容易であっても，１人１人の作曲家にとっての実用的な作曲支援のためのツールを用意することは不可能であるだろう。

1-2. IRCAMという場

1980年代の終わり頃からこのような有益なコラボレーションの場が欧米の各研究機関に現れることに従って，この作曲支援という分野の展開を見せることになる。中でも特にそれらコラボレーションが目覚ましく展開していった場としてフランス国立音響現代音楽研究所 Institute for music / acoustic research and coordination（通称 IRCAM）が挙げられる。

この IRCAM という作曲家と技術者達との交流の場は主として Instrumental Acoustics, Room Acoustics, Analysis / Synthesis, Music Representations, Real-time Applications 等といった異なった研究テーマを持つ複数のチームから成り立っている。そしてここにはフランス国内からのみならず海外からも毎年多くの作曲家，演奏家，技術者と音楽ハードウエア及びソフトウエアの開発関連企業といった人が集まる場である。

以下にこれらチームの研究活動の概要を述べる。

Instrumental Acoustics

主任研究員 René Caussé を中心には振動，共鳴および指向性を含む楽器の音響の性質を研究する。この作業はフィジカルモデリング（Modalys ソフトウェア）による音響合成のための適用，及び既存楽器の改良と製作技術の向上の可能性を提供する。

Room Acoustics

主任研究員 Olivier Warusfel を中心に空間における音の伝達を研究する。
Spatialisateur を用いて音響の様々なシミュレーションを行っている。
ここでの研究である音の空間処理は作曲家のみならず，コンサートホールの建築，オーディオ・ポストプロダクションなどにも大変有益な情報をもたらす。

Analysis / Synthesis

主任研究員，Xavier Rodet を中心に音響合成，分析そして DSP をベースにした音加工のメソード，音響情報からのアコースティック・モデリング等の研究，開発を行っている。ここで開発されたアプリケーションには Audiosculpt，Diphone Studio，Modalys などが有名である。

Music Representations

Gérard Assayag 主任研究員を中心にコンピュータによる音楽の構築を調査，研究している。注目すべき点はグラフィカルなプログラミング言語を用いた音楽の構築のための，OpenMusic

環境の開発である。

Real-time Applications

Norbert Schnell を主任研究員にこのチームでは音と音楽のリアルタイム処理のためのコンピュータシステムの企画と製作を行っている。作曲家はインターラクティブに楽器とコンピュータによる音響合成された音とを結合させた作品を創作するためにこれらのシステムを使用する。この作業は初期段階以来，IRCAM で行われてきたリアルタイム環境の研究である，音響合成と加工の研究と特にコンピュータと演奏家との同期システム（スコア追従システム）と身体表現の分析の研究からの結果として作られたアプリケーション（4X, ISPW, Max 及び jMax）によって追求する。

これらの活動が示すように IRCAM では「音」に対する研究範囲は多岐にわたっている。そして特筆すべきは，この活動形態はそれぞれのチームが相互作用によって発展を繰り返しているということである。

たとえば Instrumental Acoustics チームで研究されたデータは Music Representations チームでオーケストレーションの可能性の追求として活用されたり，また Analysis / Synthesis での SDIF という音響データのフォーマットは OpenMusic 上で非常に重要なプログラムであるというようにそれぞれのチームの技術者と作曲家の関係だけでなく，これらチーム同士のコラボレーションも密接でなくては現在の IRCAM の発展はなかっただろう。

上述に加えて，コンピュータの処理能力とユーザー・インターフェースの飛躍的向上，そして何よりもパーソナル化が進み現在では研究所レベルのコンピュータ（通称スーパーコンピュータ）と同様のパフォーマンスがラップトップコンピュータで得られるようになったこと，そしてインターネットを含むネットワークシステムが標準化されたことにより，わざわざ海外の研究所まで出向かなくても，作曲家がピアノの前に座って作曲するという従来の環境に高度な作曲支援またはコンピュータ音楽環境を簡単に追加できるようになった点もこの分野の急進的な成長を見せる理由であるだろう。

1-3. Music Representations チームによる作曲支援環境の構築

このチームの作曲支援のための主要アプリケーションである OpenMusic が開発されるまでには FORMS, PatchWork といったメインプログラム及び Esquisse，Chroma といった作曲家のプライベートプログラム等幾つかのプロセスを経て現在に至っている。OpenMusic はこれらの処理能力，適用範囲，グラフィック・インターフェース等の飛躍的な向上を遂げた拡張版であるといえるだろう。

前身となったこれらのプログラムとの共通することとして基底に Lisp という言語を採用している点が挙げられる。周知のことではあるが，コンピュータ上での言語はC，Fortran,

Pascal，Cobol，Basic，Lisp，そして近年ではプラットフォームを限定しないJAVA等の他非常に多くのものが存在している。当然これら言語はそれぞれに特徴をもっており適所で使い分けられているが，中でもC言語は親しみやすく一般的なものであり,そして何よりも処理速度の速さ等の理由から今日では様々な状況下で使用されている。ではこの作曲支援のためのプログラムがそのような利点を持ったC言語ではなくLispを長年採用しているのにはどういった意味があるのだろうか？

「LISPは人工知能の分野で使用され発展してきたが，汎用の計算機言語である。人間の知能や知性に関する研究（たとえば，心理学，認知科学，言語学）に広く使用されてきた。したがって，Lispを学習することは文字列処理と人間の知的活動の関係を知ることへの重要なプログラミングツールの獲得につながる。しかし，古典的人工知能に重要な役割を演じてきたことから認知度は低いが，Lispはアプリケーション開発言語としても利用されている。」（プログラム言語 Lisp P18 石丸清登）

C言語等の言語は命令型，または手続型に分類される言語である。それらに対してLispは関数型と言われ，その処理系等は階層リストを処理することに非常に優れているといえる。

また，CommonLispはインタープリターであり，実行する際にC言語のようにコンパイルする必要がなくプログラミングとその実行を対話のスタイルで行うことができ，なんと言っても簡単で手軽である。作曲支援という目的のために必要なのは独立したアプリケーションではなく，簡単に拡張可能な計算機である。もちろん処理能力は重要な項目であり，C言語でコンパイリングしたものには及ばないが，やはりそれを犠牲にしても処理手順とその適用範囲を優先することになる[1)]。こういった理由，特徴から作曲支援のアプリケーションにLisp（特にCommonLisp）言語をそしてプログラミング手法にCLOSを採用していくことになる。

2．OpenMusicの環境

2-1．OpenMusicの概要

OpenMusicはその前身であるPatchWorkと同様CLOSに基づくオブジェクト指向の音楽に特化された，GUI（グラフィカル・ユーザー・インターフェース）を実装したプログラム言語である。これは音楽プログラム言語として有名なMaxと表面的には類似したものに見えるかもしれない。しかし，これら2つは全く異なるコンセプトの上に成り立っているものである。Maxの詳細についてはここでは割愛させていただくが，Maxがその適用をリアルタイム環境

1) Common Lispインタープリターも年々高速化し，OpenMusicプロジェクト当初に比べると今ではその処理能力は遙かに向上している。

を構築することに対してOpenMusicは処理結果を実行という形でその都度，演算を行っていく。つまり，前者は演奏の拡張としてインターラクティブという点において最もその能力を発揮するが，後者のように膨大な数のデータを扱ったり，非常に複雑なアルゴリズムを自由に組んでいくということ等は不向きである。そして後者は音楽の１つのアイディアのレベルからより大きい時間で扱う構造の構築等，物理的な意味においての直線的時間で考えられる作曲には大変有益であるといえる。

作曲家がOpenMusicでの基本的な作業過程として，音程構造，リズム構造，フォルム等，音楽のあらゆるパラメータをコンピュータによって演算し，最終的には器楽曲等のアコースティックなものにそれらを投影するというスタイルが一般的である。

2-2. OpenMusicのフレームワーク

2-2-1. ワークスペース

OpenMusicのメイン・インターフェースはワークスペースと呼ばれるコンピュータの基本的なオペレーション・システム（OS）に似た，アイコンベースのGUIを採用している（**図2-1**）。ワークスペースではOpenMusicのオブジェクト或いはフォルダー，サブフォルダーを保存，整理しておく場所で，このプログラムを起動したときにまず最初にアクセスするところである。

また，ワークスペースはユーザー毎或いはプロジェクト毎に幾らでも作成することができ，しかもこれらはネットワーク経由でも動作し，完全に作業を独立させて使用することが可能

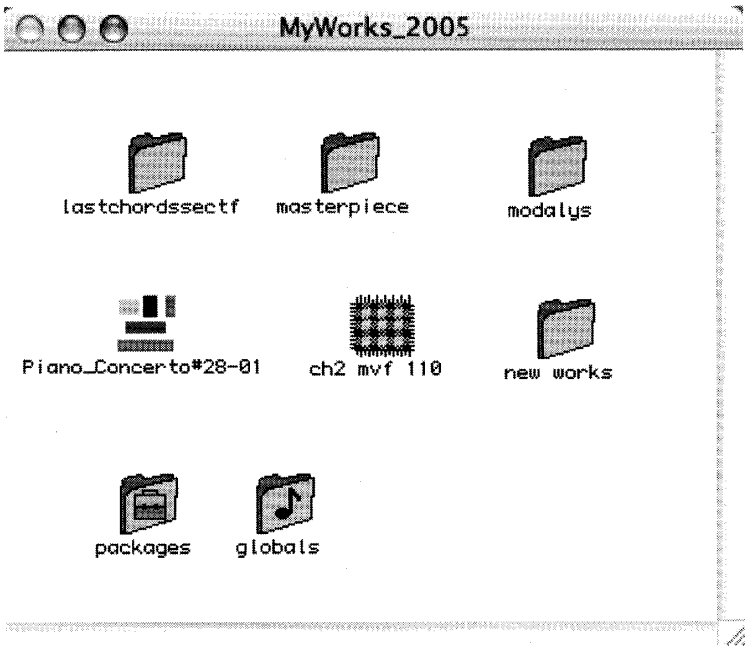


図2-1

である（**図2-2**）。

前身のPatchWorkではこのような様々なアイディアやプロジェクトを管理するというコンセプトはなく，ただ単にアルゴリズムを組むというだけのものであったことに比べるとOpenMusicにおけるこのファイル・マネージメント・システムは更に創作環境下でのコンピュータが担う範囲を大きく広げたといえる。

OpenMusicのワークスペース上で扱われるファイルは，ユーザーが作業するためのものとして，パッチ，マーケットと呼ばれる独自のファイルとそれらを整理するためのフォルダ，（**図2-3**）その他はOpenMusicのカーネルである，パッケージ，グローバルといった特別なフォルダ（**図2-4**）の計5種類から構成されている。

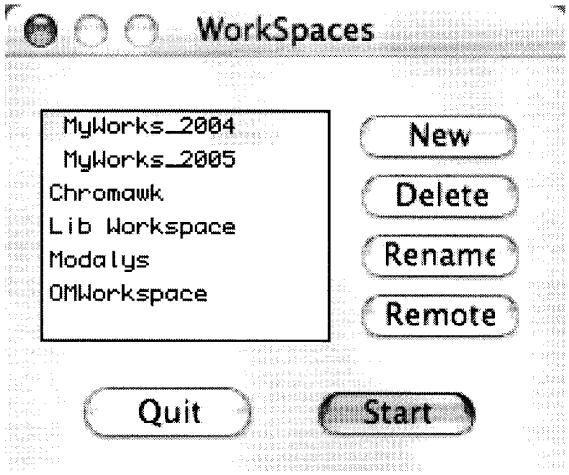


図2-2

図2-3

図2-4

2-2-1. パッチ及びモジュール

作曲家はアイディアを具現化しながら広げていくためにPatchWorkと同様に上記のパッチファイルの上で1つ1つのモジュールと呼ばれる部品をフローチャートのように関連づけて

線で繋いでいくことでプログラムを進められていく（**図2-5**）。

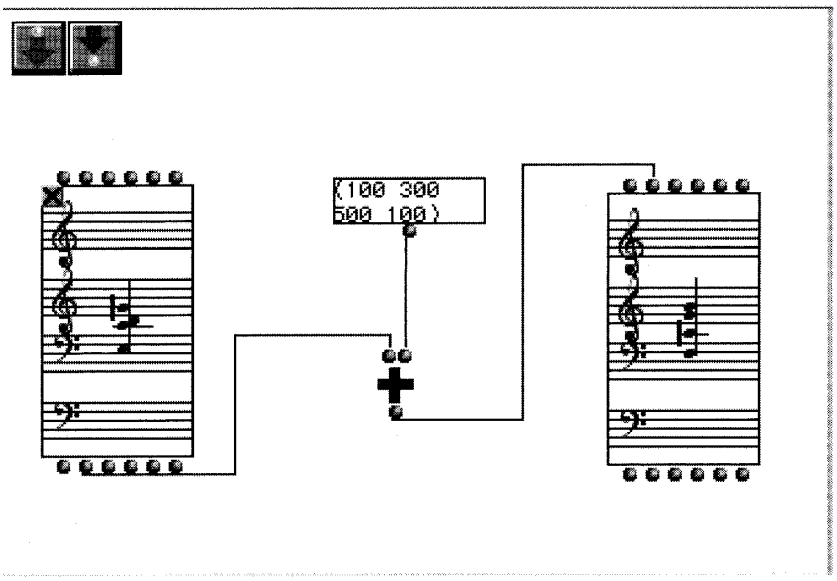
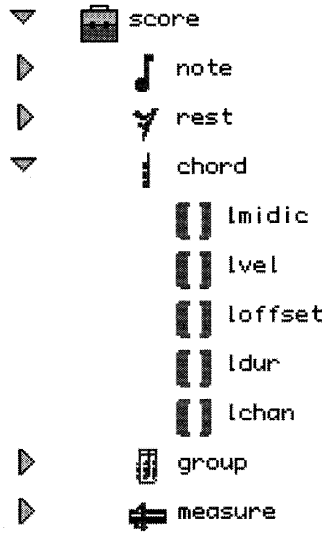


図2-5

OpenMusicには初期状態で多数の有効なモジュールが用意されており，それらは音楽に特化されたものからCommonLispのベーシックなファンクションまで様々である。またこのモジュールは作曲家自身がアブストラクションやユーザー・ファンクションとして新たに作成し追加していくことも可能である。

モジュールの構造について更に詳しく見ていくと，モジュールを形成している要素としてクラス，そしてファンクションがある。何れもCommonLispでプログラムが書かれ，コンパイルされ，高速に処理できるように計られている。クラス，ファンクションともCLOSの共通概念である基本的な定義である。オブジェクト指向のプログラミングにおいては，まず必要な事物を計算機上でソフトウエア的にモデル化する。



具体的にパッケージ，クラスとファンクションの関係を上図の例から見ていくと，定義されたクラスである note, rest, chord, group, measure は score と名付けられたパッケージと呼ばれる１つの配下に属する。そしてそれらクラスを形成するものは，複数のファンクションである。たとえば chord のクラスは，lmidic, lvel, loffset, ldur, lchan の５つのファンクションから成っている。

OpenMusic のモジュールは単純な足し算のような１つのファンクションで形成されるものと音程，音の強さ，チャンネル，デュレーション等のファンクションを複数を合わせてはじめて成立するクラスの２種類に分けられる。クラスに含まれるこれら１つ１つのファンクションも同様にアイコン化され，パッチ上に配置することが可能であるが，単体で使用するケースは非常に少ない。

2-2-1. マーケット

マーケットはパッチで作成したモジュール，パッチまたはその処理結果，MIDI ファイル，オーディオ・ファイル等を時間軸上に並べていくことで音楽の構造を自由に創ることを目的としている（**図２－６**）。これは一見シーケンス・ソフトウエアと同種のものとも思えるが，そのコンセプトは全く異なるものである。一般に市販されているシーケンスソフトの最終目的は簡単に言ってしまうとコンピュータによる自動演奏である。そこでは MIDI 或いはオーディオ・ファイルのリージョンを置き換えまたはエフェクト処理等を施すといった作曲と呼べるような行為はあるにせよ，このシーケンスソフト上で考え創作するというのには限界があるだろう。しかし，マーケット上で配置したものは音楽の進行と同時に再処理，演算を行うことが可能なのである。つまり，このマーケット上でパッチを作成し，その処理のタイミン

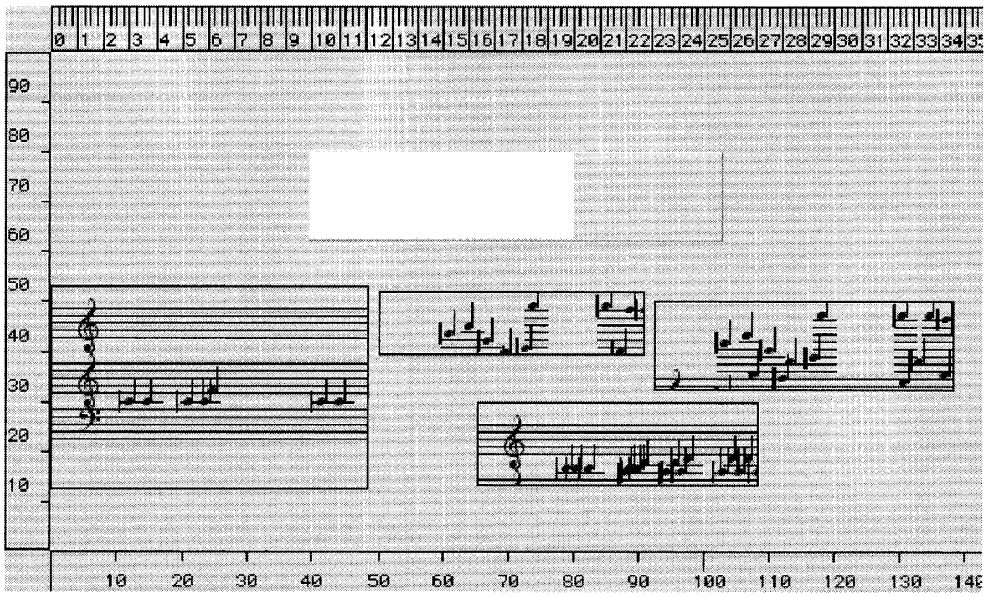


図２－６

グを時間軸上で制御するといったいわばイベント・シーケンサーとでも言うべきものである。これは例えば1つのリージョンを確率や推計学等のメソードを用いての作曲においては，その出力の結果は毎回違った不確定なものとなるが，マーケットを使うことによりこのような場合にも適応できるようになっている。

横軸は拡大縮小可能な時間軸で縦軸はフリーでありトラックレスとして使用する。
更にここで作成したものの結果はDIGIDESIGN社のPROTOOLSにomptというライブラリーを用いてエクスポートすることができ，電子音楽を作曲する際には非常に便利である。また，MIDIの部分だけをFINALEMUSIC社の楽譜作成ソフトFINALEのENIGMAファイルとしてエクスポートすることもできる[2)]。

2-3. OpenMusicのコンフィギュレーション

以下に開発環境，技術使用等コンフィギュレーションを記す。

使用環境

Apple Power Macintosh, OS 8以上及びOSX[3)]，64 MB以上のRAM，120 MB以上のHD.

企画及び開発

Gérard Assayag，Carlos Agon（IRCAM Musical Representationsチーム）
Olivier Delerue（コラボレーター）

音楽専門技術アドバイザー

M. Andreatta，J. Baboni，J. Fineberg，K. Haddad，C. Malherbe，M. Malt，T. Murail，O. Sandred，M. Stroppa，H. Tutschku

パートナー企業及び研究機関

Digitool（MCL），GRAME（MidiShare），FINALEMUSIC（Finale）

---

2) IRCAMの微分音テンプレートを用いればFINALE上で1/4，1/8音の表現が可能となっている。
3) 2005年9月現在，既にOS 9での開発は完了している。この論文はOSX（10.3.8）を用い，CommonLisp 5.0（OSX native）及び，OpenMusicのバージョンは4.9.1の環境を前提にしている。
その他，UNIXバージョン（linax）と現在開発中のWindowsXPバージョンがある。

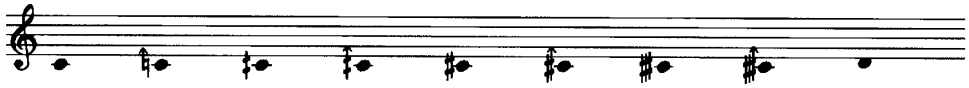
3．OpenMusic における作曲支援

3-1．記譜システムとリズムの表記

OpenMusic に用意されている音楽オブジェクトの中でも特徴的なことの１つは，アイディアの入力から出力まで五線を用いて行えることである。これは特にコンピュータの知識を持たない作曲家でも簡単にアクセスでき，感覚的に作業を進めることを可能にしたといえる。これに加えてすべての表示した音楽オブジェクトを微分音，グリッサンド，音色の変化等を含め外部/内部音源によるプレイバックができることも挙げられる。このプレイバックについては市販のシンセサイザーのレベルから IRCAM で製作されたアンサンブル・アンタルコンタンポランの奏者による１音ずつ採取のハイクオリティサンプルまで，作曲家の機材環境によってはかなりのハイレベルなシミュレーションが行える。

では具体的に OpenMusic のシステムを見ていきたいと思う。

OpenMusic での音程の表現は MIDI セントを使用する。これは聞き慣れなれないものかもしれないが，通常の MIDI の数値に100を掛けたものである。これは微分音のプレイバック及び表示するためである。

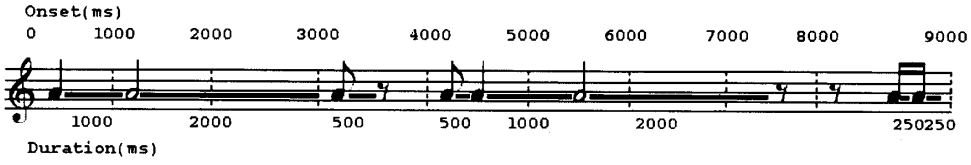


midics :	6000	6025	6050	6075	6100	6125	6150	6175	6200
	+ 0	+ 1/8	+ 1/4	+ 3/8	+ 1/2	+ 5/8	+ 3/4	+ 7/8	+ 1

次に音価についてであるが，OpenMusic では２種類の音価の表現があり，モジュールによって使い分けられている。それらは note（単音），chord（和音），chord-seq（和音を含む単一トラック），multi-seq（複数トラック）のそれぞれのモジュールはプロポーショナルな記譜を voice（単一トラック），poly（複数トラック）では定量記譜を採用している。

プロポーショナルな記譜ではオンセットタイムとデュレーションタイムという考え方で音価を確定する。これら２つともミリ秒単位で表現され，つまり１秒は1000ミリ秒ということになる。オンセットタイムとは時間軸上で０秒からのイベントの時間であり，デュレーションタイムは１音１音のそれぞれの長さである。

以下にそれらオンセットタイムとデュレーションタイムの関係を音符で示すと，



従って，オンセットタイムのリストは（0 1000 3000 4000 4500 5500 8500 8750），対するデュレーションタイムは（1000 2000 500 500 1000 2000 250 250）と指定し，この記譜システム上での時間の表現はこの2つのリストが必須となる。

定量記譜の場合はリズム・ストラクチャーを以下の公式に従って，数値リストの形で与える。

(小節数（小節のリスト))

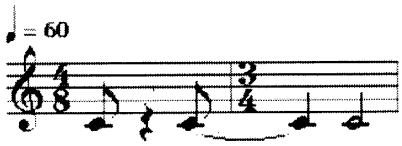
小節数＝オプション

小節のリスト＝((拍子)(リズム比率のリスト))

拍子=例えば4//4又は（4 4）

リズム比率のリスト＝((パルスの数)(パルスのリスト))

例えば，（？(((4 8)((4（1 -2 1))))((3 4)((3（1.0 2))))))）のリスト[4]から出来るリズム・ストラクチャーは，以下のようになる。



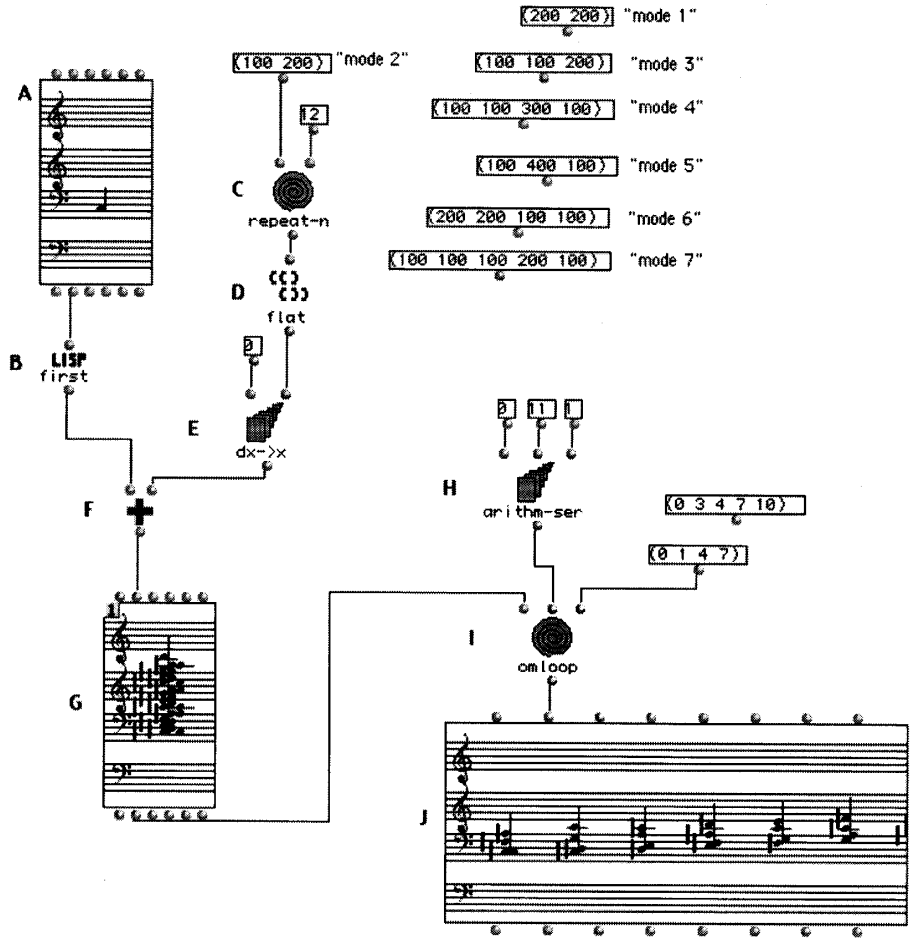
前章でFINALEへの書き出しについて触れたが，FINALEのENIGMAファイルへの変換はこの定量記譜を採用したモジュールからしか行えない。もしプロポーショナルな記譜を使うモジュールから書き出したいときはクオンタイズすることで可能となる。

OpenMusicを使用して作業を有効にそして円滑に進めるためには，基本的な次の3つの作業の手順を頭に入れて置かなければならない。それは，何を入力するか？，どのような処理をするか？，何を出力するか？　である。当然のことに思えるが，3点を明確に計画しなければ，成り行き或いは偶然まかせのものとなりがちである。

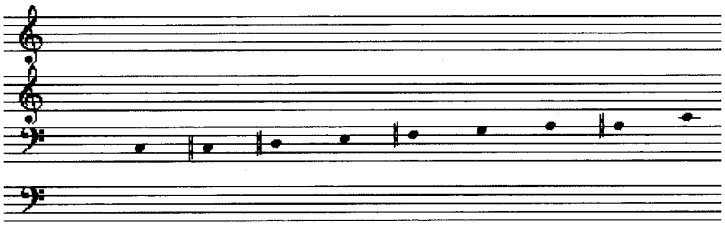
では詳しく作曲家たち自身の作曲システムをOpenMusicを用いて作業をして見ていきたいと思う。

4)　-2は休符，1.0はタイ（連結される後の音符に指定する）を表す。

3-2. O・メシアン：移調の限られた旅法



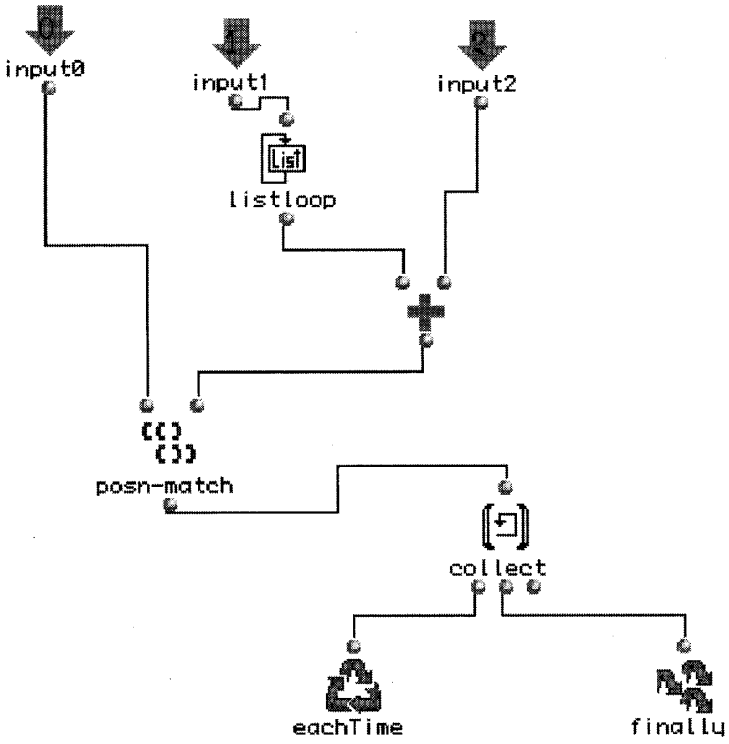
このパッチではモードの作成及びそのモードを基礎とする和音の作成を目的としている。まずモードの最初となる音の入力(A)リストからナンバーを抽出する。(B)各モードのための音の並びのデータを半音（100)，全音（200)，半音＋全音（300)，全音＋全音（400）の4種類を用いてあらかじめ作成する。それらの繰り返し回数を決定して(C)，同階層のリストにする。(D)次にそのリストを音間隔として dx->x モジュール(E)でセリーを作る。最後に入力した音とリストの足し算(F)をして指定したモードを作成する。



第２旋法の作成（G）

次に作成したモードを使用してコードを作成する。

任意の和音構成音をリストで作成する。例えば（０１４７）の場合は旋法の第1, 2, 5, 7音となる[5]。モジュール arithm-ser は数列を生成するものである。３つのインレットを持ち，１番目は生成する数列のはじまりの数，２番目は終わりの数，３番目は数列のステップを入力する。(H)この例での結果は（0 1 2 3 4 5 6 7 8 9 10 11）という単純な12個の数列ができる。これらをomloop モジュール(I)に入力して最後の処理をする。

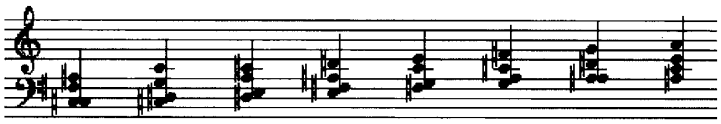


この omloop モジュールは OpneMusic のモジュールのなかでも非常に多機能で，重要なものであるが，プログラミング経験のない人にとっては少し難解かもしれない。

最初に Input 1 と Input 2 の部分を見ていきたいと思う。Input 1 は先程 arithm-ser で生成した数列が入力される（0 1 2 3 4 5 6 7 8 9 10 11)。Input 2 には指定した和音構成音のリストが入力される。入力された Input 1 からこの omloop 内でのみ使用できる特別なモジュールの１つ

5)　CommonLisp では１番目の要素は１ではなく０である。

listloopに入力されているが，このモジュールの役割は入力されたリストの中の数値を1度に全リストの要素ではなく，最初の要素（数）から1個ずつ次に接続するモジュールom+に受け渡していくというものである。つまりこの場合は（0+(0 1 4 7), 1+(0 1 4 7), 2+(0 1 4 7), …… 11+(0 1 4 7)）ということになり，((0 1 4 7)(1 2 5 8)(2 3 6 9)(3 4 7 10)(4 5 8 11)(5 6 9 12)(6 7 10 13)(7 8 11 14)(8 9 12 15)(9 10 13 16)(10 11 14 17)(11 12 15 18)) となる。次にposn-matchモジュールにInput 0とこの処理したリストを入力して処理するわけだが，このモジュールは1番目のインレットに素材となるリストを入力し，2番目のインレットには1番目のリストから何番目の要素を抽出かを決定するための数またはリストを入力する。先程処理したリストを2番目に入力することでposn-matchモジュールでは ((0 1 4 7)(1 2 5 8)(2 3 6 9)…(11 12 15 18)) を ((0番目 1番目 4番目 7番目)(1番目 2番目 5番目 8番目)…(11番目 12番目 15番目 18番目)) というような役割となり，1番目での旋法リストから該当する要素（音）を抽出していていく。



3-3．T・ミュライユ：スペクトル音楽～周波数変調モデル

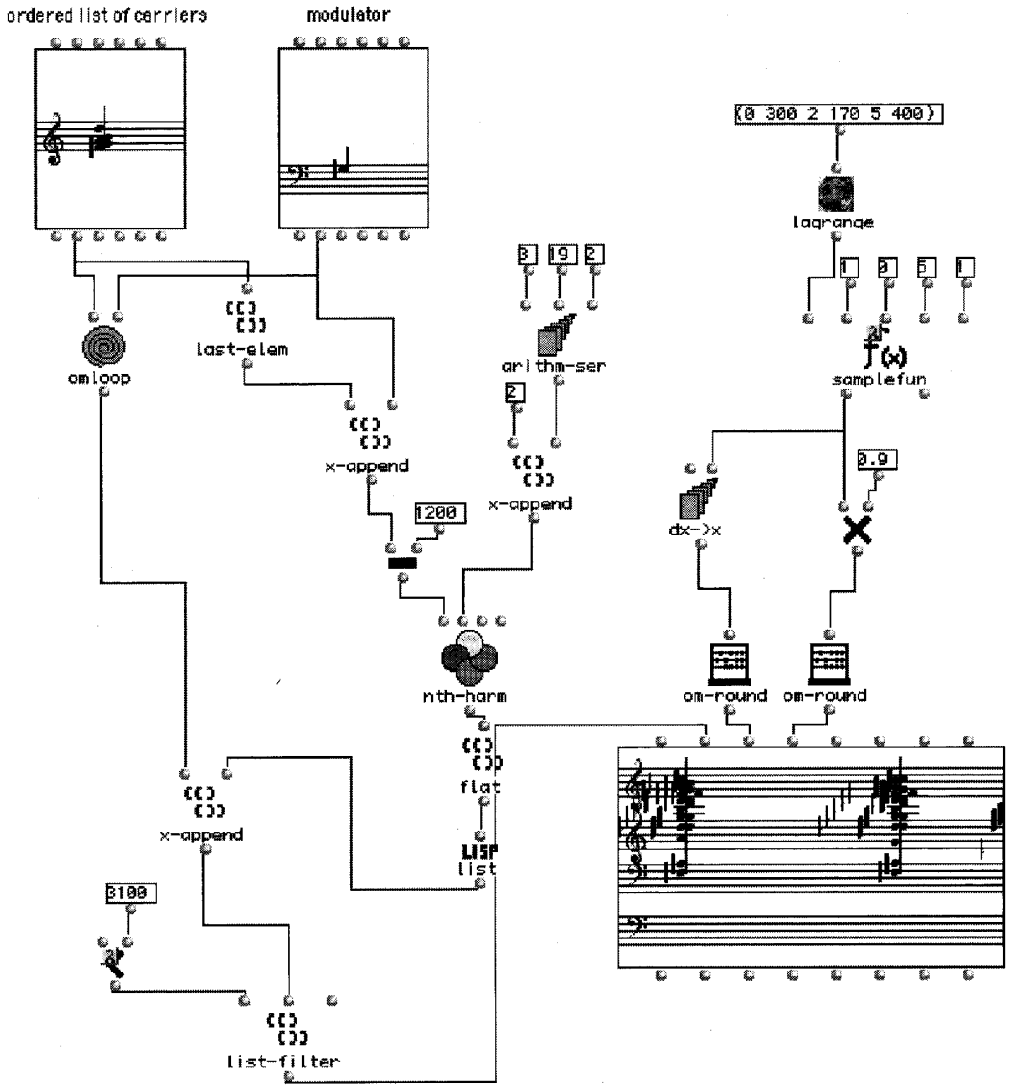
フランスを代表する作曲家，Tristan MURAIL（1947－）は作曲支援の分野に早くから関わりをもっていた作曲家の1人である。PatchWorkやOpenMusicのような統合的なプログラムが出来る前から自身の作曲システムのためのプログラムを開発していた。その後それらを単体としてではなくPatchWork，OpenMusicの追加機能としてユーザー・ライブラリー（Esquisse）という形で活用することになる。このEsquisseライブラリーはスペクトル音楽またはスペクトル楽派と呼ばれるT. MURAIL, G. GRISEY, M. LEVINAS, H. DUFORTたちの作曲システムのためのツールのコレクションの一部でもあり，大変興味深いものである。数ある彼らの作品の中でも様々な場面での作曲支援のツールの使用が見受けられる，オーケストラのためのGondowna（1980）から最初のセクションで使われるベルサウンドの周波数変調のシミュレーションをOpenMusicを用いて見ていきたいと思う。

この作品の作曲システムを理解するにはまず，周波数変調の知識が必要となる。周波数変調においてはキャリア，モジュレータという2種の役割を持たせた，正弦波オシレータを用いてそれらを掛け合わすことで加算合成では得ることの出来ない，高次倍音を含んだ音響を作り出すという理論である。周波数変調は以下の公式に従って演算される。

C+(M*I), C-(M*I)

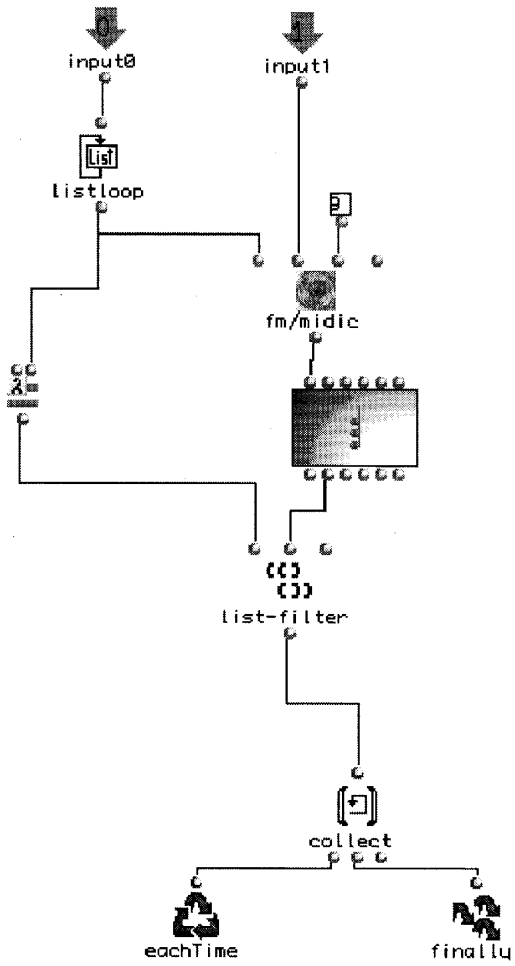
C＝キャリア，M＝モジュレータ，I＝変調指数（1, 2, 3, 4…）[6]

6) 例えば変調指数が3の時：C+(M*1), C+(M*2), C+(M*3), C−(M*1), C−(M*2), C−(M*3) となる。そしてそれらの結果は全て絶対値で処理する。



このパッチでは最初の音（和音）と最後の和音のシーケンス以外は全てMIDIセントではなく周波数を使用して演算する。

キャリア（6700 6900 7100 7400 6600），モジュレータ（5600）がomloopモジュールに入力され，周波数変調及びその処理結果のフィルタリング（0を取り除くため）がループ処理され，合計9つの和音を生成する。



その結果は ((7436 5394 7950 1814 8348 5784 8670 6894 8940 7566 9176 8048 9382 8424 9566 8734 9734 8996) (7570 5794 8050 1914 8428 5380 8736 6694 8998 7432 9226 7948 9426 8344 9606 8668 9770 8940) (7708 6156 8156 3918 8512 4776 8808 6436 9060 7268 9280 7828 9476 8250 9650 8590 9810 8872) (7924 6644 8326 5274 8652 2548 8926 5874 9162 6942 9370 7598 9556 8072 9724 8444 9878 8750) (7372 5174 7902 2964 8310 5942 8638 6980 8914 7624 9152 8092 9362 8460 9548 8764 9716 9022)) となる。

6番目の和音を金管楽器に近い倍音列（2 3 5 7 9 11 13 15 17 19）をつくる。次にそれらに基音を与えて倍音列を作る。基音に関しては，キャリアの最後の要素とモジュレータのリストから（6600 5600）オクターブを下げたもの ((om-(6600 5600) 1200)) を使用する。(5400 4400) そしてその2つの倍音列を組み合わせたものを和音とする。結果は，（6600 7302 8186 8768 9204 9552 9840 10088 10304 10498 5600 6302 7186 7768 8204 8552 8840 9088 9304 9498）となり，先程の9つの和音の最後に連結する。((7436 5394 7950 1814 8348 5784 8670 6894 8940 7566 9176 8048 9382 8424 9566 8734 9734 8996) (7570 5794 8050 1914 8428 5380 8736 6694 8998 7432 9226 7948 9426 8344 9606 8668 9770 8940) (7708 6156 8156 3918 8512 4776 8808 6436 9060 7268

9280 7828 9476 8250 9650 8590 9810 8872) (7924 6644 8326 5274 8652 2548 8926 5874 9162 6942 9370 7598 9556 8072 9724 8444 9878 8750) (7372 5174 7902 2964 8310 5942 8638 6980 8914 7624 9152 8092 9362 8460 9548 8764 9716 9022) (6600 7302 8186 8768 9204 9552 9840 10088 10304 10498 5600 6302 7186 7768 8204 8552 8840 9088 9304 9498))

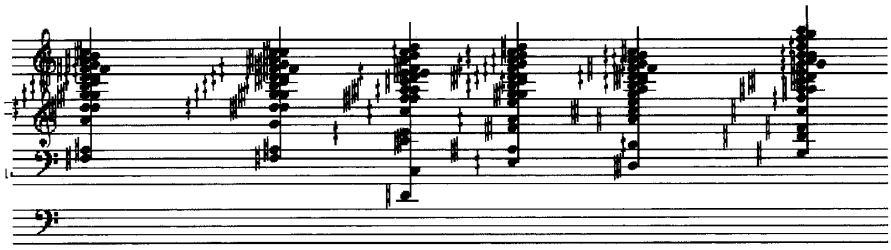
次にこの和音シーケンスから（3100）以下を取り除くためにlist-filterモジュールを用いてフィルタリング処理を行う。つまり，この作品は最終的には生楽器（オーケストラ）で演奏するためピアノの最低音の2度下のG音以下はあっても表現出来ないためである。((7436 5394 7950 8348 5784 8670 6894 8940 7566 9176 8048 9382 8424 9566 8734 9734 8996) (7570 5794 8050 8428 5380 8736 6694 8998 7432 9226 7948 9426 8344 9606 8668 9770 8940) (7708 6156 8156 3918 8512 4776 8808 6436 9060 7268 9280 7828 9476 8250 9650 8590 9810 8872) (7924 6644 8326 5274 8652 8926 5874 9162 6942 9370 7598 9556 8072 9724 8444 9878 8750) (7372 5174 7902 8310 5942 8638 6980 8914 7624 9152 8092 9362 8460 9548 8764 9716 9022) (6600 7302 8186 8768 9204 9552 9840 10088 10304 10498 5600 6302 7186 7768 8204 8552 8840 9088 9304 9498)) 以上で和音の構成音は完成である。

後はそれぞれの和音のリズムを決めるため，デュレーションとオンセットタイムをlagrangeモジュールで定義した関数[7]により作り出されている。

オンセット＝(0 3000 5067 6767 8667 11333 15333)

デュレーション＝(2700 1860 1530 1710 2400 3600)

オンセットはdx->xで（0 3000 (3000＋6200/3) …）のオンセットの書式に従ったリストを，デュレーションはsample-funで評価した値に0.9倍したものを用いている。



GONDWNAの最初のセクションの和音とリズム（プロポーショナルな記譜による）

前述の作曲家自身のライブラリーやある作曲家のシステムに基づいたライブラリーは数多く存在する。例えばブライアン・ファニホウはIRCAMでコンポーザー・イン・レジデンスを務めたとき，ミカエル・モルト氏の協力のもと作成されたCombineというライブラリーがある。このライブラリーは彼の作曲システムを分析するためとしても興味深いものである。

---

7) ラグランジュ補間法を用いて（0 300 2 170 5 400）の補間を演算し，その評価値をsample-funで合計6つの数値のサンプルを求める（3000 6200/3 1700 1900 8000/3 4000）。

その他イアニス・クセナキスの確率・推計学による作曲システムのAREAライブラリー（M. モルト製作）があるが，これはクセナキス自身が開発したプログラムより遙かに直感的で使いやすいものとなっている。

以下にすべての利用可能なライブラリーの一部を記す。

IRCAMパブリック・ライブラリー（IRCAMユーザーグループ内での公開）

- OMSDIF　SDIFフォーマットファイルの読み書きのための。
- OM2Csound　Csoundのスコアファイルを生成のための。
- OM2AS　AudioSculptのパラメータ・ファイルの生成のための。
- OM2Modalys　Modalysのためのフィジカル・モデリングの楽器の合成と制御のための。
- OMLZ　統計的なモデルとインプロビゼーションの生成のための。
- OMKant　リズムの定量化と分割。
- OMRC　リズム構造のプログラミング。
- OMCS　一般的な構造のプログラミング。
- OMSituation　音程，リズム構造のプログラミング。
- Repmus　Music Representationチームによるモジュール集。
- Profile　旋律の輪郭と曲線の操作のための。
- Morphologie　数値または記号の連鎖のファンクション集。
- OMChaos　カオス理論に基づくモジュール集。
- OMAlea　確率・推計学に基づくモジュール集。
- Esquisse　スペクトル音楽のための。
- OMMiel　楽曲分析のための。
- OM2pt　マーケットからPROTOOLSのセッション・ファイルへ書き出しのための。

IRCAMプライベート・ライブラリー（非公開）

- OM-Tristan：　Esquisseの機能拡張ファンクションとT. ミュライユ自身の作曲システム，シンセサイザーのコントロール，Max，CSoundとの連携等
- Karim：　K. ハダッドのパブリック・ライブラリーの追加ファンクション集

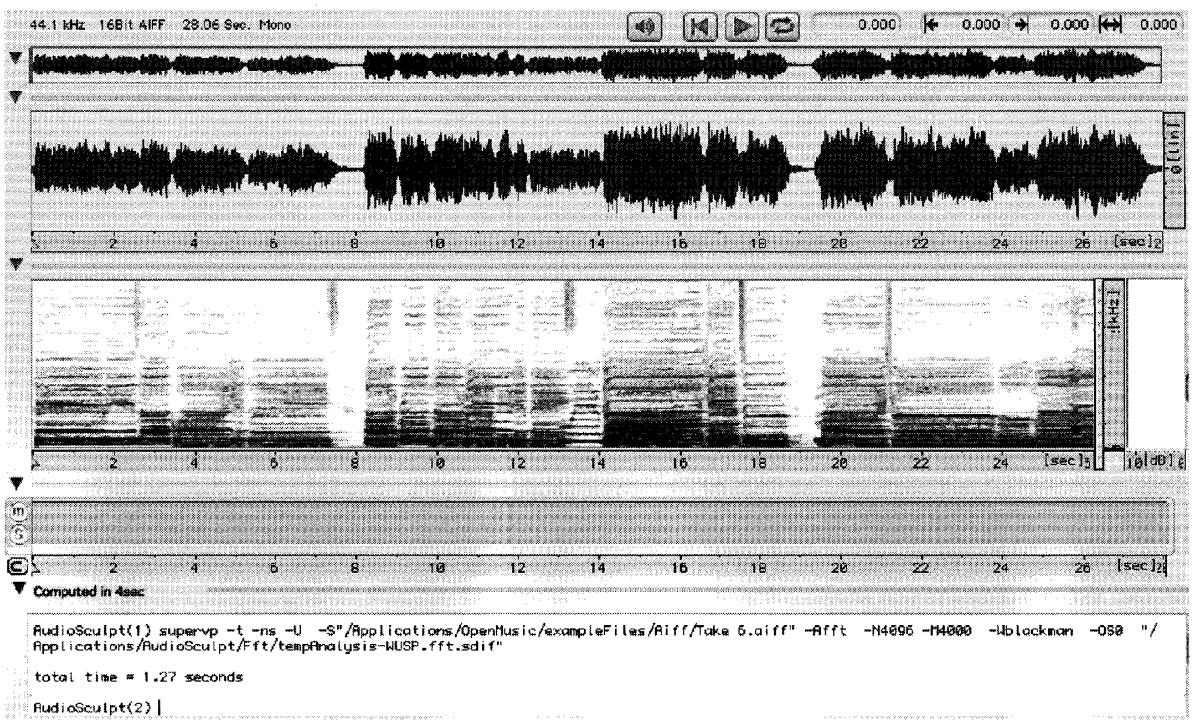
4．OpenMusic と音響分析，合成

この分野のプロジェクトは当初，CPU の性能の問題から計算機としてだけに留まっていた。従って，OpenMusic は音響をダイレクトに扱って音響生成をするのではなく，データを作成する役割であり，それらをエクスポートするという作業が通常であった。しかし近年の凄まじい CPU の高速化とともにその役割も変化を見せることになる。つまり実際に OpenMusic 上で音響を生成することが可能となった。もちろん現在でもそのデータ作成の役割も非常に有用なものであることには違いはない。

ではそれらの使用について 2 つに分けて見ていきたいと思う。

まず従来から行われてきたデータのインポートとエクスポートについてであるが，この作業は他のアプリケーションの補助的なものとしての援用である。具体的には AudioSculpt（音響分析と再合成のためのツール），Diphone（サウンド・モーフィングを行うためのツール）等 IRCAM の他ソフトウエア，電子／コンピュータ音楽では非常に有名な MIT の CSound 等との連携である。

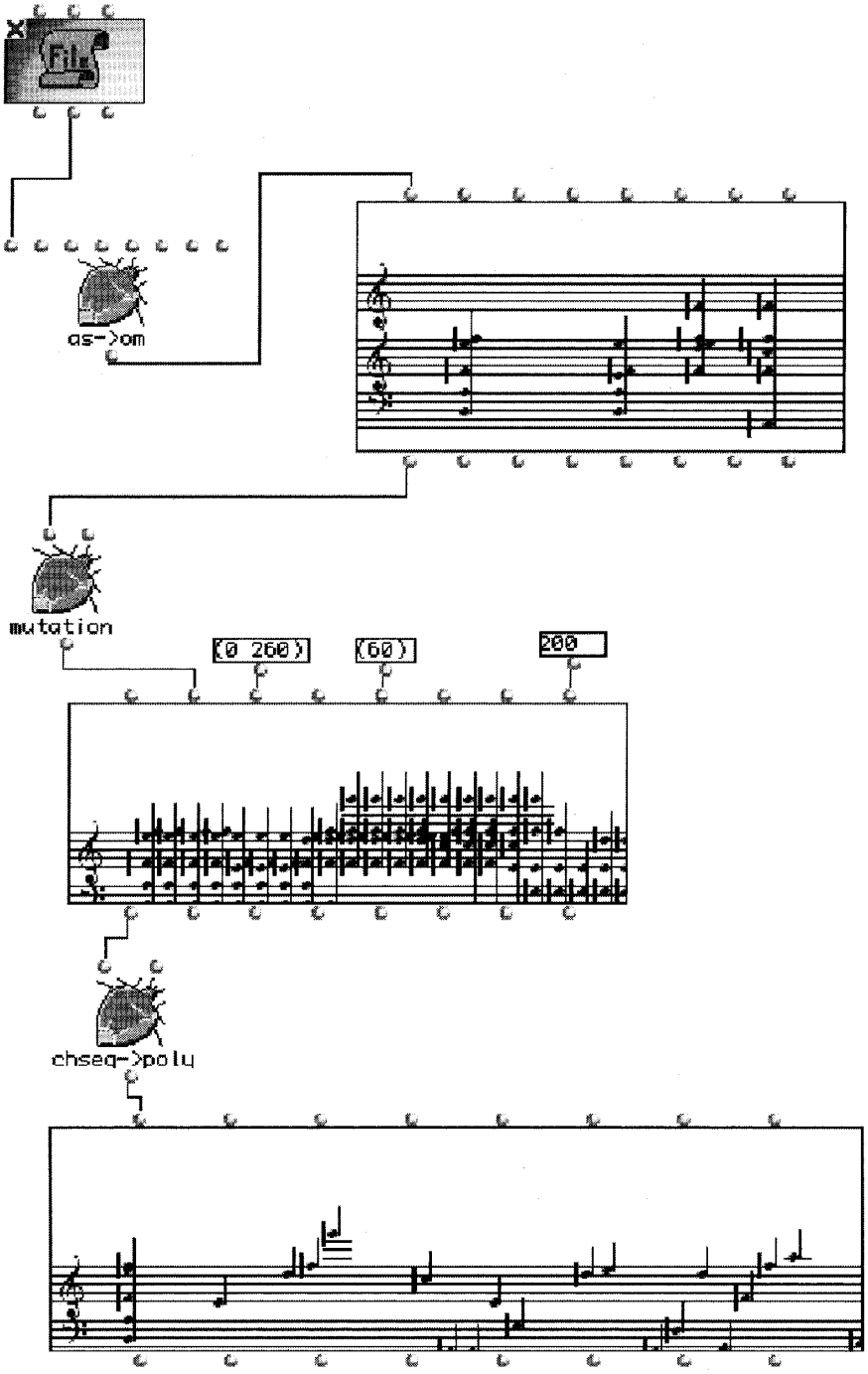
たとえば om-repmus というライブラリーを用いて AudioSculpt で高速フーリエ変換等の音響分析したデータを OpenMusic に読み込ませてアコースティック楽器のための作曲のマテリアルにするという前章で述べたスペクトル楽派の作曲家達が頻繁に行っていた手法である。



AudioSclpt を用いて高速フーリエ変換

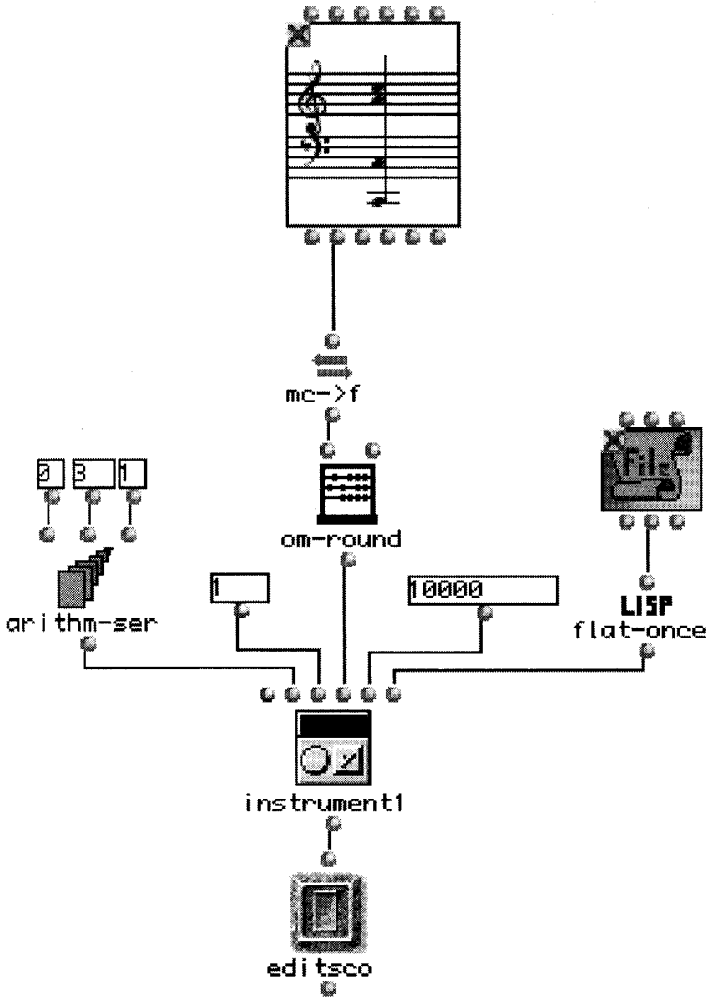
```
( PARTIALS    194
( POINTS      2
   0.350          146.558          2.581
   2.073          146.558          2.581
)
( POINTS      2
   0.350          220.746         -1.485
   2.073          220.746         -1.485
)
( POINTS      2
   0.350          328.508         -5.500
   2.073          328.508         -5.500
)
( POINTS      2
   0.350          371.268          1.013
   2.073          371.268          1.013
)
( POINTS      2
   0.350          441.431         -7.107
   2.073          441.431         -7.107
)
( POINTS      2
   0.350          662.056         -2.381
   2.073          662.056         -2.381
)
( POINTS      2
   0.350          743.076          1.527
   2.073          743.076          1.527
)
( POINTS      2
   2.085          147.079          2.200
   2.965          147.079          2.200
)
( POINTS      2
   2.085          220.753         -0.437
   2.965          220.753         -0.437
)
( POINTS      2
   2.085          328.489         -4.186
   2.965          328.489         -4.186
\
```

分析データをテキストファイルでエクスポート



1段目：解析したFFTデータをMIDI値に変換，2段目：0.26秒ごとの音の変化を表示（ミューテーション），
3段目：2段目で作成したものを素材として新たに作曲

CSoundとの連携については，OM2Csoundライブラリーを用いてスコアファイルを作成することが可能である。




```
;p1     p2      p3      p4          p5      p6      p7      p8      p9      p10     p11     p12
i1      0       1       65.4060 10000   0       -6      -32     -20     -50     800     1150
i1      1       1       130.8130        10000   0       -12     -26     -25     -44     270
i1      2       1       440.0000        10000   0       -16     -35     -40     -60     325
i1      3       1       523.2510        10000   0       -20     -15     -40     -56     350
e
```

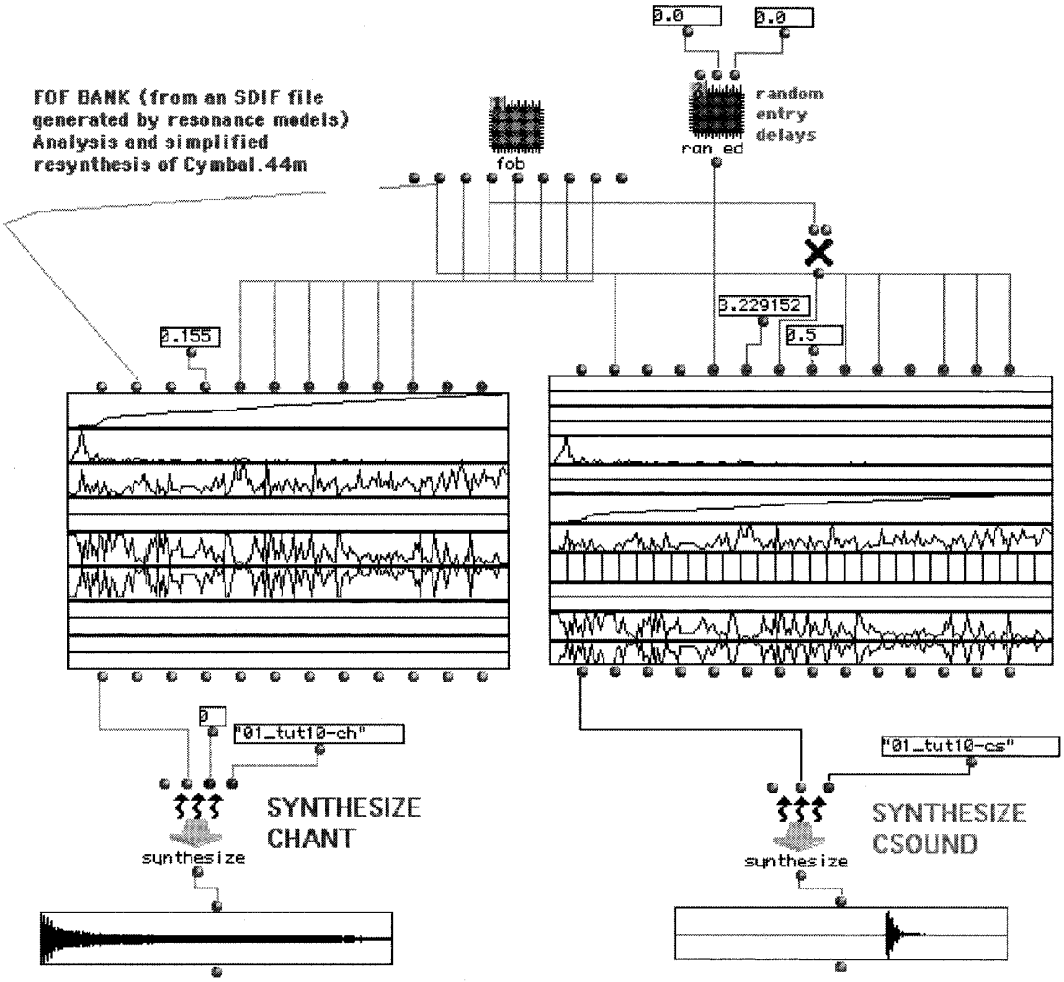
OpenMusicのアウトプットをCSoundのスコアファイルとして書き出すプロセスの例

何れにしてもOpenMusicのGUIを用いて直感的に行えること，そして何よりパッチ上ではユーザー・ライブラリーを含め，全てのOpenMusicで提供されている機能が利用できるわけであり，例えばある作成したパッチのアウトプットをCSoundのスコアファイル或いはAudioSculptのパラメータ・データと絡めることが容易にできることは非常に有益だといえるだろう。

次にOpenMusic上で音響生成を行っていくことの例としては，特にOMChromaとOM-Modalysという2つのライブラリーが挙げられる。この2つとも近年，IRCAMとCMNATで積極的に研究されている音情報のフォーマットであるSDIFを利用する。OMChromaは 作曲

家マルク・ストロッパが自身のための音響生成プログラム Chroma シンセサイザーを1985年に開発したものを OpenMusic ライブラリーへ移植したバージョンである。このライブラリーで扱うことの出来る音響合成のためのエンジンは Chant と CSound である。前述の CSound を扱うライブラリーではあくまで CSound の書式に沿ったスコアファイル生成することを目的としていたことに対して，Chrome 上では CSound のエンジンを用いて OpenMusic 独自の書式（パッチ）で音響を生成していくものである。当然それらは OpenMusic の追加モジュールとしてアイコンベースで扱える。

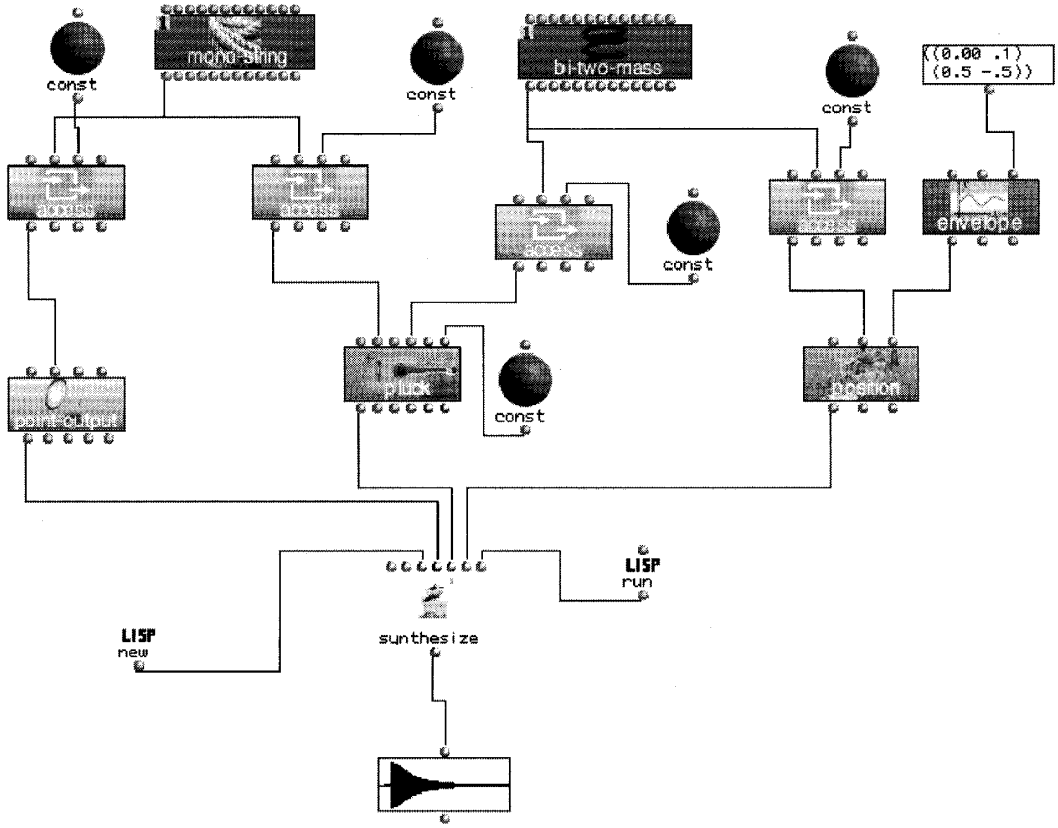
Chant は IRCAM で1979年に開発されたフォルマント・シンセシスのための強力なプログラム[8] であるが，このプログラムは Diphne，Max / MSP といった他のアプリケーション上でも可動するものである。OpenMusic 上での用法は CSound と全く同じである。（作業のプロセスという意味においては）



OMChroma でシンバルの音を 2 つのシンセシス・メソード（Chant，CSound）で再合成するプロセスの例

8) このプログラムで生成された作品としては湯浅譲二の＜世阿弥九位＞4 ch. テープと室内オーケストラのための（1989）が有名である。この時は OpenMusic / chroma によるものではなくて，VAX 11 / 780を使用していた。

またOM-ModalysはIRCAMで1982年に開発されたフィジカル・モデリング・シンセシスのためのプログラムModalys（当時の名称はMosaic）をOpenMusicライブラリーへ移植したバージョンである。ModalysもCLOSをベースとして作成されているので連携がスムーズである。このプログラムもCSound同様テキストベースで作成したものから音響生成のコンパイリングを行うというもので使い勝手に問題があったため，これをアイコンベースで扱うためのアプリケーション，Modaly-serやMax[9]のプログラムを使ってコントロールしていたなど，これまで様々な過程を経て現在のかたちとなった。



OMModalysで撥弦仮想楽器のプログラミングの例

5．OpenMusic / CLOS統合環境としての今後の発展

ここまで作曲支援のツールをOpenMusicに限定して述べてきたが，OpenMusicというプログラムは単体のソフトウエア＝アプリケーションとして考えるのは非常に難しい。このプログラムはあくまでCommonLispの上でインタープリタとして動作しているのであると考えた方が容易である。従って，表面的なOpenMusicの見た目は1種のGUIあるいはWindowsシステムに過ぎないといえる。しかしこのことはOpenMusic或いはCLOSでの作曲環境の発展

9) 現在もMax / MSP上でModalysのリアルタイム操作のためのRMI-Modalys，Max@Modalys（いずれもIRCAM Max Library）が活躍している。

の可能性を遮らない理由の１つである。つまり通常のコンピュータのアプリケーションのバージョンアップは，その意図はほぼ開発者とその周辺の人間に依るものに限られている。決まり切ったことをするだけのものであればむしろこの方が都合がよいのかもしれない。しかしこれまで述べてきたように作曲支援や音響合成の分野はユーザーがその発展に参加できることがもっとも必要であるのではないだろうか。それが証拠に現在では OpenMusic のカーネルのソースコードは定められた条件の下，無料で配布されている。世界中のユーザーは CommonLisp を用いて自分の作曲システムをモジュール化し，更にそれらをライブラリーとして作成することが非常に簡単に出来るようになっている。また，研究所レベルにおいてはスタンフォード大学で開発されたアルゴリズミック・コンポジションのためのツール CommonMusic 及び音響合成のための CommonLispMusic のファンクションはすべて OpenMusic で読み込むことができるようになり，パッチ上で本機能と同様にモジュールの単位で扱うことが可能となった。CLOS をベースに考えられたその他のプログラムやライブラリーは簡単に OpenMusic の機能として追加拡張することができるまでに発展してきた。現在 OpenMusic カーネルは，更なる高速化，リアルタイム環境での使用，既存アプリケーションとの更なる強力な連携，プラットフォームの拡張等の開発プロジェクトが進行中である。OpenMusic は統合環境として他に類を見ない，より強力で有益なものになっていくだろう。

6．作曲家と作曲支援ツールの今後

「創作においては，プログラムと作曲家による相互作用はさほど信じていません。が，プログラムの把握，つまり作曲するための方法をよく知っていれば，作品を表現するための手段として大いに助けになると考えます。ある知り合いの作曲家がこう言っていました。“ある人があることへの言い方を知っているとする。しかし，それはその人が結局，何を言いたいか知っていることとは別のことである”…あることを表すにはまずアイディアが必要で，次にそれを音楽で実現する方法を知らなければなりません。その知識は表現をするための方法となります。しかし，技術的なことを忘れて表現するためのツールを自由に操るには，やはり技術を十分に心得ていなければなりません。」（Sound & Recording Magazin /後藤英・M. モルトのインタビュー記事より）

これまで述べてきたように，OpenMusic は多様化する音楽或いは作曲スタイル等，様々な状況下において比較的簡単に論理化し，適応していくキャパシティーを持ち合わせている。しかしその反面，時にその便利さ故，作曲家と OpenMusic との関係を誤ってしまう危険性も同時に持ち合わせているといえる。OpenMusic はスイッチ１つ押せばオートマティックに作曲をしてくれるものではないし，そのコンセプトもない。また，過去の偉大な先人たちの作曲システムを単純にそのまま自分の作曲に流用する目的でもない。（楽曲分析の目的での使用

は大変有効であるが）しかし皮肉なことにそれらも可能性として簡単に出来てしまうのである。M・モルト氏が語るように作曲家のアイディアとテクニックのバランスをよく認識することが重要である。そして言うまでもないことであるが，創作するのはコンピュータではなく，あくまで人間側であるということを忘れてはいけない。

作曲家がこのツールに本当の意味での有益を求めるために，我々が得たこの新しいテクノロジーである，OpenMusic を含めた作曲支援のツールとの距離を十分に考えていくこともこの分野の技術的な発展と同時に大きな課題であるのではないだろうか。

（本学講師＝作曲担当）

参考文献

石丸，清登：
2001 『プログラミング言語 Lisp ―入門からマルチメディアまで』（東京：ASCII）

HADDAD, Karim：
2003 OpenMusic Tutorial（PARIS：IRCAM）

後藤，英：
2001 世界のリゾーム化…？ ヨーロッパ最先端 Computer Music 情報 Sound&Recording Magazin（東京：リットーミュージック）

ASSAYAG, Gérard RUEDA, Camilo LAURSON, Mikael AGON, Carlos DELERUE, Oliver：
1999 Computer Assisted Composition at Ircam : PatchWork & OpenMusic.
Computer Music Journal 23 : 3（LONDON : ）

ASSAYAG, Gérard：
1998 Computer Assisted Composition Today
1st SYMPOSIUM ON MUSIC AND COMPUTERS.
"Applications on Contemporary Music Creation ,Esthetic and Technical Aspects". CORFU, 23-24-25 October 1998.

DIGITOOL 社
2001 Getting Started with Macintosh Common Lisp

AGON, Carlos STROPPA, Marco ASSAYAG, Gérard：
2000 High Level Musical Control of Synthesis in OpenMusic（Berlin : ICMC 2000）