音楽情報科学 30－1
（1999．5．29）

ポピュラー音楽のアドリブ生成システムBoP3の実装

岸田良朗*、大林幹生†、林恒俊‡
立命館大学

概要

当研究は、ロック等のポピュラー音楽におけるアドリブを自動生成するプログラムBoP3の開発をパーソナルコンピュータ上で行うものである。コンピュータ上のGUIや、コンピュータに接続されたMIDI楽器を用いて演奏家のフレーズデータを収集し、そのデータを材料として、楽曲の伴奏に合わせてアドリブを自動生成するプログラムである。各フレーズデータは伴奏との関係を表す属性値を持ち、楽曲の進行とともにその属性値を基にした計算を行なってフレーズを選出していくことにより、アドリブの自動生成を行なっている。

Implementation of Automatic Generation of Improvisation in Popular Music

Yoshiro Kishida, Mikio Obayashi, Tsunetoshi Hayashi
Ritsumeikan University

Abstract

We have been developing a program which automatically generates improvisations in popular music on a personal computer such as Apple Macintosh. We call our developing project *Boogie Project 3 (BoP3)*. BoP3 generates improvisations by simulating the improvisation process of actual musicians. BoP3 gathers phrases those are exploited when generating improvisations. Improvisation is generated reffering to information flow of an accompaniment tune.

1 はじめに

ロック等のポピュラー音楽において、アドリブとかインプロヴィゼイションと呼ばれる即興演奏は重要な意味を持つ。ジャズなどのようにインプロヴィゼイションそのものが演奏活動となる場合もあるし、音楽製作の場においてはアイデア創出や動機の探索のためのセッションな

*kishida@hayalab.cs.ritsumei.ac.jp
†baya@hayalab.cs.ritsumei.ac.jp
‡hayashi@hayalab.cs.ritsumei.ac.jp

どとして行なわれることも多い。本研究では、このような軽音楽における即興演奏を、コンピュータによって自動的に生成するプログラムの開発を行なっている。

もっとも、作曲や即興演奏などの創作活動は本質的には高度なヒトの感性や能力を以てしてのみなし得るものであり、コンピュータがその代わりをするのは不可能であると言っても過言ではないと考える。コンピュータによる音楽の自動生成は、そのような創作過程をある程度現象論的にとらえ、擬似的なシミュレイションとして行なわせるのであれば可能なのではないかと考え、これまでコンピュータによる即興演奏の自動生成を行なうプログラムの開発を行なって来た [1] [2] [3] [4]。プロジェクト名を *Boogie Project* とし、開発が第 3 段階に入っていると言う意味でヴァージョンを 3 として、プロジェクト、あるいはプログラム名を、略して、以下、*"BoP3"* と呼ぶこととする。

2 アドリブ自動生成プロジェクト BoP3

本プログラム BoP3 は、一般に入手可能なパソコン上で動くアプリケーションとして開発する。使用するパソコンは、Apple 社の Macintosh シリーズである。BoP3 においては、音楽情報の取り扱いをすべて MIDI によって行なうため、Macintosh の MIDI システムとして Opcode 社の *OMS (Open Music System)* を使う。また、伴奏を行なうために、同社の音楽情報の統合ソフトである *Vision 3.5* を用いる。

2.1 BoP3 におけるアドリブの定義

アドリブは、貯えられた断片的なフレーズを楽曲の流れにあわせて次々とくり出して行くことで行なわれると仮定する。本研究において、このようなフレーズをアドリブ・フレーズと呼ぶこととする。そして、アドリブは、与えられた楽曲を伴奏として、伴奏が演奏されるのと同時に即興的にアドリブ・フレーズを順序良くり出して行くことであると定義する。BoP3 で即興演奏を行なわせるためには、最初にアドリブ・フレーズを収集することから始めなければならない。アドリブ・フレーズを集めたものを BoP3 ではプレイヤー (演奏家) と呼ぶこととする。

さらに伴奏となる楽曲を用意し、そのコード進行などの音楽的に必要な情報を用意する。伴奏となる楽曲の条件として、

- カウントできる脈動的なリズムがあり、
- 調性が明確であること

が必要である。

2.2 BoP3 の仕様

全体は大きく三つの部分からなる。一つはアプリケーション本体で、全体を統括し、MIDI 情報を扱うための基盤部分とアドリブの生成機関を持つ。二つ目は、演奏家として働く部分で、アドリブ・フレーズとその属性情報を貯える。三つ目は、伴奏を受け持つ部分であるが、実際に演奏はせずに伴奏楽曲の情報を扱う。

2.2.1 アプリケーション本体部分

アプリケーション全体を統括し、MIDI 情報を扱うために OMS と通信を行なうための基盤を持ち、アドリブ生成を行なう。実体としては、メニューのアプリケーションに関わるコマンド群と、MIDI の操作のためのフローティング・ウィンドウや、音符情報表示のためのピアノロール・ウィンドウ等が主たるものである。

コントロール・ウィンドウ コントロール・ウィンドウは、レコード・ボタンや再生ボタン、そしてガイドの音源の切り替えのためのポップアップ・メニューなどが置かれ、MIDI 情報の録音、再生などの制御を行なう。コントロール・ウィンドウはフローティング・ウィンドウになっている。

フレーズ・ウィンドウ　フレーズ・ウィンドウは、MIDI 録音された音符情報を表示し、編集するためのウィンドウである。レコードされたMIDI 情報はピアノロール状のウィンドウに表示される。マウスのクリックにより、アドリブフレーズとして必要な音符を選択できる。

2.2.2　演奏家としてのプレイヤー部

演奏家を表すプレイヤー・ウィンドウと、アドリブ・フレーズとその属性値の入力用のダイアログから成る。プレイヤー・ウィンドウは、入力したアドリブ・フレーズを貯える。また、ファイルとして保存したり、保存されたファイルを開いたりする役目を果たす。

アドリブ・フレーズの登録は、第2.2.1節で述べたフレーズ・ウィンドウにおいてマウスのクリックなどの方法により選択された音符情報の配列オブジェクトを持って来ることで行なう。プレイヤー・ウィンドウ上のテーブルに名前が表示される。この時、ダイアログ上でその属性値を編集することができる。属性値は、以下のものである。

- そのフレーズの名称、
- 調性、
- 長さ、
- 表拍で始まる場合と裏拍で始まる場合における、様々なコードとの相性、
- 小節中における、フレーズ開始のビート位置の好き嫌い、
- くり返しの好き嫌い、
- アドリブの終わり部分の向き不向き。

2.2.3　伴奏としてのチューン部

伴奏となる楽曲の情報を扱う部分で、拍子やコード情報を編集、記憶し、アドリブ生成時にはこの情報が参照される。チューン・ウィンドウは、アドリブの伴奏となる楽曲のコード進行を表示する。マウス操作でコード・ボタンを選ぶことにより、そのコードが楽曲に1小節ずつ追加されて行く。

2.3　アドリブの伴奏

本プロジェクトでは、アドリブは、伴奏となる楽曲の上で演奏されることとする。伴奏は、BoP3 で演奏させるように実装することも可能であるが、市販のソフトウェアに高機能なものがあり、アプリケーション間で同期を取ることが可能であれば役割を分担させることができるので、外部のアプリケーションを用いる方が有利な事が多いであろうと考えた。

Opcode 社の Vision 3.5 は、広く一般に用いられているシーケンサー・ソフトで、非常に高機能である。また、*SMF* (標準 MIDI ファイル) も再生することができるので、広く流布しているSMF を伴奏に用いることができ、好都合である。Vision 3.5 は、OMS のアプリケーション間通信用のインターフェイスを用いることによって若干のアプリケーション間通信を行なう能力を持っている。このため、BoP3 と同期をとって楽曲を演奏させることができるので、これを伴奏用に用いることができる。

3　開発環境

Macintosh で動作するプログラムの開発環境としては、*Metrowerks* 社の *CodeWarrior* があり、本研究ではこれを用いる。CodeWarrior のアプリケーション・フレイムワークとしてクラス・ライブラリの *"PowerPlant"* がある。CodeWarrior、あるいは PowerPlant は、Macintosh 上での開発の事実上の標準の開発環境となっており、アップデイトも頻繁で信頼がおける開発システムである。メイリングリストなどのコミュニティも活発で、開発者同士の交流も盛んであるため情報量も多い。本研究においても PowerPlant を用いることによって、ウィンドウ表示やダイアログ表示、マウスによるボタン操作などの GUI の充実をはかり、信頼性が高く、ユーザに分りやすいアプリケーション・プログラムを構築することをポリシーとする。

OMS に関しては、Opcode 社の web ページで SDK が公開されており、ライブラリや、イ

ンターフェイスやドキュメントが用意されているので、これらをもとにMIDIまわりに関する実装を行なうことができる。

4 BoP3の実装

PowerPlantのクラス・ライブラリを母体とし、BoP3にユニークな部分のために独自のクラスの実装や、PowerPlantの派生クラスを加えてプログラムの開発を行なう。

以下では、通常のPowerPlantのGUIに関する実装以外の、BoP3独自の音楽情報に関わるMIDI回りの実装部分と、MIDIレコーダーに関する部分の実装について説明する。アドリブ生成を行なうクラスについては第5章で述べる。

4.1 OMSとMIDIまわりの実装

本プロジェクト独自のクラス群として、MIDI関係のクラス群の実装を行なって来た。これらのMIDIまわりのクラス群は、他のMIDI情報を扱うプログラムにおいても使用できるように留意した。また、PowerPlantのクラス・ライブラリへの依存をなるべく小さくするようにした。

4.1.1 MIDI基地クラス

アプリケーションとOMSとの間をとりもってOMSと直接メッセージを交換するクラスを、MIDIの基地とするという意味で*COMSMIDIBase*と名付け、実装した。MIDI基地クラスCOMSMIDIBaseはOMSの関数をラッピングして引数の処理を軽くし、プロジェクト全体からのMIDI操作を容易にしている。COMSMIDIBaseには汎用性を持たせて、他のプロジェクトにおいても使えるように考慮したが、PowerPlantの配列クラスは内部で使用するために、常に必須である。

コンストラクターと初期化 アプリケーションはスタート時にCOMSMIDIBaseのオブジェクトを生成し、直後にMIDIまわりの初期化を行なって、関数呼び出しの際の引数の数の軽減を行なっている。コンストラクターでは、グラニュラリティー(音符の粒度)の設定が行なわれる。初期化として行なわれることは、

- OMSにサイン・インを行なう、
- OMSの*Studio Setup*のカレント・セットアップより、MIDI入出力のノードを得る、
- OMSのタイマーにアプリケーションの登録を行なう、

などである。タイマーとは、*OMS Timing*のことで、OMSの時間管理に関するライブラリとインターフェイスを提供している。

コールバック・ルーチン OMSには、3種のコールバック・ルーチンを登録しておかなければならない。

ReadHook 入力されたMIDI信号を受け取るのためのコールバック・ルーチン、

AppHook OMS全般のメッセージを受け取るためのコールバック・ルーチン、

MessageCallback Timerからのメッセージを受け取るためのコールバック・ルーチン。

COMSMIDIBaseにおいて、デフォウルトのコールバック・ルーチンが用意される。ReadHookは、通常、アプリケーションは、その目的のために独自のルーチンを用意しなければならないであろう。

その他のメソッド その他に、以下のようなメソッドの実装を行なった。

コールバック・ルーチン設定メソッド コールバック・ルーチンを、アプリケーションなどで別に定義されたスタティック関数に再設定するためのメソッド。

MIDIパケットの送出メソッド 通常のチャネル・ヴォイスメッセージや、ランニング・ステイタス方式、IACドライバー向けの送出など、目的別のメソッド。

タイマー関係メソッド タイマーのスタート、ストップなど。

フレーズの登録メソッド タイマーにスタティックなMIDIパケット送出メソッドを、時間を指定してコールバック・ルーチンとして登録することにより、フレーズを演奏する。

その他 緊急時のオール・ノウツ・オフや、ピッチベンドの初期化、OMSのスタジオ・セットアップ・コマンドなど。

4.1.2 シーケンス・クラス

MIDIイヴェントの配列クラスで、フレーズ、すなわち音符データのシーケンスを扱うクラスである。配列クラスのオブジェクトに貯えられた未整理な音符データの配列を、時刻順にソートされたものに変換する。また、同時刻の時間情報を持ったノート・オン、オフ情報は、ランニングステイタスとして一つのパケットに格納しておく。

レコーダーや、ドンカマ[1]を用意するためのクラスは、このクラスを継承することによって、OMS Timingに引き渡す音符シーケンス(フレーズ)を容易に作ることができる。

4.2 レコーダー部

MIDI録音されたフレーズの登録を行なうために、市販のシーケンサー・ソフトに多く見られるようなピアノ・ロール状のMIDI情報表示ウィンドウを実装し、編集できるようにした。

受け取ったMIDIパケットはReadHookにより即座にスタティックな配列に格納され、アプリケーションの通常のループの中で、逐次、配列クラスのオブジェクトに再格納されて、データとして貯えられる。この配列オブジェクトが、ウィンドウ上への表示や、編集、再生、ファイルへの保存、そして、アドリブ・フレーズの登録のために使われる。貯えられるパケットの種類は、ノート・オン、オフとピッチベンド情報である。不要なピッチベンド情報を記録しないようにするオプションを設けて、録音されるMIDIシーケンスから、アナログ楽器において発生する音程の不安定な部分を取り除くことができるようにした。ただし、ギターのチョーキングやバイブレイション等の、奏法に関する必要な情報を漏らさないために、捨て去るピッチベンド情報の大きさに、適当な閾値を設定する。

5 アドリブの自動生成

アドリブの自動生成は、アドリブ生成クラスのオブジェクトが行なう。BoP3はアドリブ開始のコマンドを受け取ると、アドリブ生成クラスにアドリブ開始のメッセージを送り、アドリブ開始とともに伴奏を演奏させる。アドリブ生成クラスは、楽曲の進行状況に応じてアドリブ・フレーズを選択し、選ばれたフレーズを、順次演奏させる。

5.1 アドリブ・フレーズの選択方法

アドリブ・フレーズの選択を行なうには、まず乱数を用いて選択するための「的」を用意し、各フレーズに、ある大きさの乱数の「的」の一部分の領域を割り当てておいて、発生させた乱数がどの領域に入ったかを調べてフレーズを決定するという方法を用いる。それぞれのアドリブ・フレーズの、的の中で占める領域の大きさは、その時点におけるそのフレーズの「出現意志」の大きさの2乗に比例した大きさとする。

5.2 アドリブ・フレーズの出現意志

各アドリブ・フレーズの選出され得る確率を決定するために、前節で述べたアドリブ・フレーズの「出現意志」を定義する。これは、アドリブ・フレーズのある時点での出現したいという意志を数値として表すもので、その値は第2.2.2節で述べられた、各フレーズの「コードとの相性」を基に計算される。

[1] レコーディングの際のテンポを一定に保つためのガイド音

5.2.1 楽曲の流れにおけるアドリブ・フレーズの出現意志

まず、出現意志を決定するために、その時点でのコードと、フレーズが演奏される拍のウラ、オモテ位置を調べて、そのコードとの相性を得る。その値を基に、小節中のビート位置の好き嫌いの程度による相性への修飾を加えて、出現意志を計算する。さらに前のフレーズとの音程関係のつながりを自然にするために、同じ音程や遠く離れた音程のフレーズの出現意志を低く押さえるように修飾を加える。

5.2.2 その他の要素のアドリブ・フレーズの出現意志への影響

そのアドリブ・フレーズ演奏中にコードの変化が見込まれる場合は、コードとの相性をより繊細に判断しなければならない。またアドリブの終わりが近付いている場合は、そのフレーズがアドリブの終わりに向いているかどうかの判断を行なう。前に選ばれているフレーズがくり返しを好むフレーズである場合は、回数を記録し、適当な回数である間はそのフレーズの出現意志を大きくしてやる必要がある。

5.2.3 アドリブ・フレーズのグループ化

アドリブ・フレーズの中には、小節の中の位置によって非常に出現意志が高くなる傾向のあるフレーズの存在が考えられる。このようなフレーズにとっては、楽曲がその位置に来る前に前もって出現を予約させておいた方が良い場合がある。そのため、楽曲の進行に先立って、2拍、あるいは4拍先あたりにフレーズの立候補がないかどうかを調べる。ある場合には、さらに前後につながる相性の良いフレーズを探してグループ化し、そのグループ化されたフレーズを新しく一つのアドリブ・フレーズとしてその出現意志を高くしてやる。また、グループ化されたフレーズが複数になる場合は、それぞれに高い出現意志を持たせて競争をさせる。

6 結論

BoP3に、伴奏にあわせてアドリブを演奏させてみると、かなり自然な流れを感じさせる自動生成を行なう。

ユーザー自らがアドリブ・フレーズを入力できるため、ユーザー自身がコンピュータに入り込んだような印象を感じることができる。今のところ試用してもらった人数は少数であるためはっきりした評価はできないが、教え込まれた演奏以上のことをやってみせるというような創造的な即興演奏ができているとまでは言えないかもしれない。

現在の仕様では、伴奏のために高価なシーケンサー・ソフトを使う必要がある。しかしながら、より簡単な条件でBoP3を使用するためにはBoP3自体に伴奏の機能を持たせることも検討しなければならない。

今後の発展として、アドリブ生成時に実際の人間との共演を実現し、その演奏とのインタラクティブな交流を行なうアドリブ自動生成の実装を行なう考えである。

参考文献

[1] 岸田良朗: “知識情報に基づく, 軽音楽におけるアドリブ演奏の自動生成”, 第50回(平成7年前期) 情報処理学会全国大会講演論文集 (分冊1), 1995.

[2] 岸田良朗、林恒俊: “軽音楽におけるアドリブ演奏の計算機による自動生成プログラムの実装”, 情報処理学会第52会全国大会(平成8年前期), 1996.

[3] 岸田良朗、林恒俊: “アドリブ演奏の自動生成プログラムの実装におけるフレーズ選択について”, 情報処理学会第53回全国大会 (平成8年後期), 1996.

[4] Yoshiro Kishida and Tsunetoshi Hayashi: “Implementation of Automatic Ad Lib Generating Program in Popular Music”, Proceedings of International Conference on Computer Music & Music Society Oct. 15-19, 1996, Shanghai China, 1996.