# 軽音楽における即興演奏の自動生成システム

岸田良朗*、大林幹生†、林恒俊‡
立命館大学

**概 要**

当研究は、ポップスやロックなどの軽音楽における即興演奏をパーソナルコンピュータ上で自動生成させるプログラムの開発を行うものである。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、そのデータを即興演奏の材料として用いて自動生成を行なうシステムである。各フレーズ・データはそのフレーズの性質を表す属性値を持っている。楽曲の進行とともにその属性値を参照し、よりその音楽的流れにマッチした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う。

# Automatic System 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. 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. Improvisations are generated reffering to the relationship between the nature of each pharase and the flow of the accompaniment tune.

## 1 はじめに

音楽情報のコンピュータによる処理の分野において、芸術として、また技術として多彩な研究がなされ、大きな成果が得られて来ている [1] [2]。本研究は、軽音楽における即興演奏をコンピュータによって自動生成させることを目的としている。

ポップスやロックなどのポピュラー音楽において、アドリブ、あるいは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っている。ジャズにおいては、演奏活動の中心はインプロヴィゼイションそのものである場合が多いということは良く知られている。また、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どとして行なわれることも多い。

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

作曲や即興演奏などの芸術的な創作活動は、本質的には高度な人間の感性や能力を以てしてのみなし得るものであり、軽々にコンピュータが人間に代わって創造活動を行なうことが可能であると考えることはできない。ただし、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て、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 [3] [4] [5]。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 即興演奏自動生成システム

## 2.1 軽音楽における即興演奏の定義

軽音楽における即興演奏は様々な方法によって行なわれる。例えばジャズなどにおいては、テーマの旋律からモチーフを得て、それを変形、発展させて行くというようなアドリブの方法がある。また、偶発的な音なども排除せず、次々に新しい音を探し出して自由に演奏するというような即興演奏もある。

本プロジェクトにおいては、アドリブは、演奏家が貯え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。そして、そのような断片的なフレーズをアドリブ・フレーズと呼ぶこととする。つまり、本プロジェクトにおいて、アドリブは、与えられた楽曲を伴奏として、同時に即興的にアドリブ・フレーズを選択し、順次演奏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。アドリブ・フレーズは、前もって用意しておくことも可能であるが、ユーザーが演奏家であれば、自ら自分のアドリブ・フレーズを登録することができる。

BoP3 におけるアドリブ・フレーズの長さは、8分音符を単位とする。したがって、アドリブの進行も、常に8分音符を単位として考えることになる。

アドリブ・フレーズの長さは、8分音符1個のものから1小節を越えるような長いものまで考えられるが、8分音符3個分を中心にして2個から4個分の長さを基準とする。

## 2.2 即興演奏の伴奏

また、伴奏となる楽曲 (チューンと呼ぶ) が必要であるが、その楽曲のコード進行などの音楽的な情報を用意しなければならない。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、
- アドリブの開始時と終了時が明確であること

が必要である。

カウントできる脈動的なリズムとして具体的には拍を考え、1拍は4分音符としてカウントすることとする。

## 2.3 即興演奏の自動生成

アドリブの生成は、プレイヤー (演奏家) と、チューン (楽曲) のドキュメント・ファイルが用意された状態で開始される。アドリブ生成の命令によって伴奏の演奏が開始され、同時にアドリブ・フレーズの中から適当なフレーズが順次選びだされて、アドリブが進んで行く。

実際に人がアドリブを行なう場合、演奏家が次々にフレーズを選びながらアドリブを進行させるのであるが、自動生成システムにおいては、それぞれのアドリブ・フレーズ自身が、その時点において出現したいとか、あまり現れたくないというような、「出現意志」を持っていると考える。出現意志は楽曲の進行とともに変化するため、それが出現確率の変化として現れ、アドリブに多様な変化をもたらす。

## 2.4 アドリブ・フレーズの選択方法

アドリブ・フレーズの選択は、最終的には乱数を用いて行なう。乱数を用いて選択するためには、発生させる乱数の範囲の大きさの「的」を用意し、各フレーズに、ある大きさ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の割合は、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乗に比例した割合になるように割り当てる。

## 2.5 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、アドリブ・フレーズの出現意志を定義する。その値は第3.2節で述べられる、各フレーズの「コードとの相性」を基に計算される。

### 2.5.1 楽曲の流れにおけるアドリブ・フレーズの出現意志

楽曲の進行に従ってコードは変化する。その時点のコードと各アドリブ・フレーズの相性を調べるのであるが、拍をオモテとウラに8分音符で分けて、そのどちらの位置からそのフレーズが始まるかによってもコードとの相性が違っている可能性があるため、その時点が拍のオモテであるのか、ウラであるのかを調べて、コードとの相性の値を得るようにする。

その値を元に、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高関係のつながりを自然にするために、前フレーズの最後の音高と次候補のフレーズの最初の音高との差が、2度音程ないし4度音程程度のフレーズよりも、同じ音高や遠く離れた音高差であるような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。

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。アドリブは、コードの変化を遅れて感じて演奏されると不自然になる場合が多いため、実際のコードの変化を先読みして、次に来るコードに合ったフレーズを選択できるようにする。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適し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ある程度大きくしてやる。

### 2.5.2 アドリブ・フレーズの予約選択

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拍先あたりまでにフレーズの立候補がないかどうかを調べ、もしあれば、そこで現在選ばれているフレーズと再度競争を行なって予約フレーズを優先させるかどうかを決定する。

先立ってフレーズが予約された場合は、その時点から予約されている時点までの拍数に合うようなフレーズを探し、その出現意志が高くなるようにする。

# 3 即興演奏自動生成システム BoP3の概要

本システム 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。

BoP3は大きく分けると、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 MIDI 基地とアドリブの生成機関を持つ。二つ目は、演奏家としての仕事をする部分で、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、実際に演奏をするのではなく、伴奏楽曲のコード進行などの音楽の流れとしての情報を扱う。

## 3.1 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 MIDI 基地となるオブジェクトを持つ。また、アドリブ生成を行なうためのアドリブ生成クラスのオブジェクトを持つ。アプリケーション部のユーザー・インターフェイスは、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状のウィンドウなどが主たるものである。

### 3.1.1 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、そしてテンポをガイドするリズム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、ユーザーから常にアクセス可能なフローティング・ウィンドウとなっている (図 1)。

### 3.1.2 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 (図 1)。MIDI 録音された 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する。マウスのクリックにより、音符を入力することも可能である。

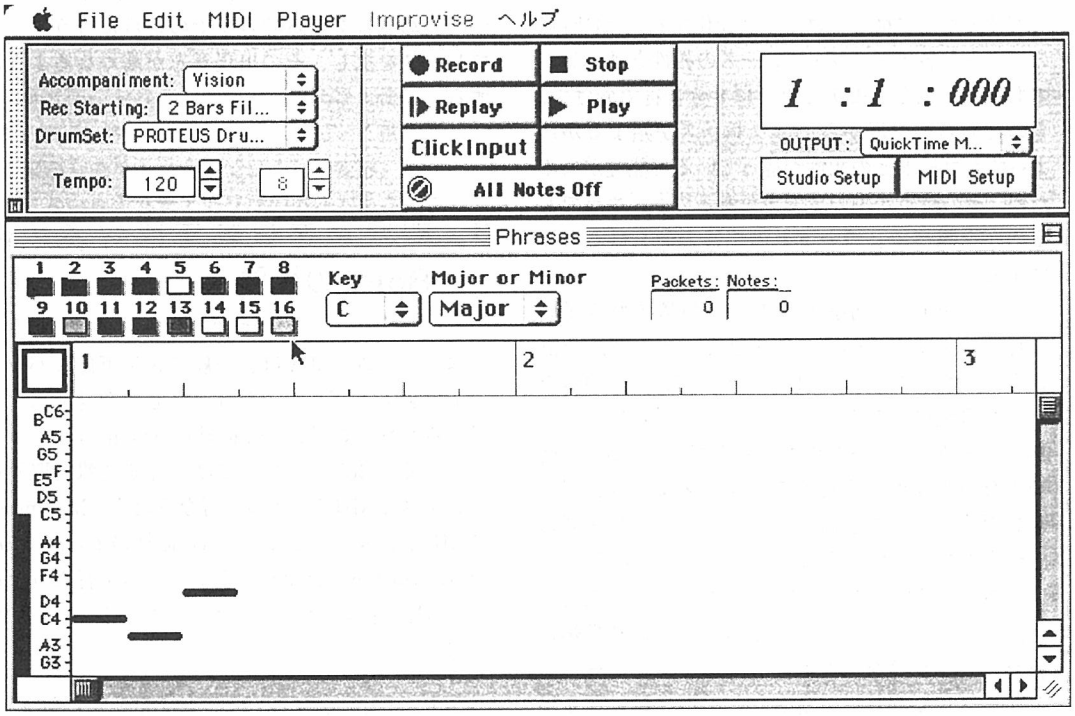


図 1: コントロール・ウィンドウとフレーズ・ウィンドウ

## 3.2 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 (図 2、図 3)。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

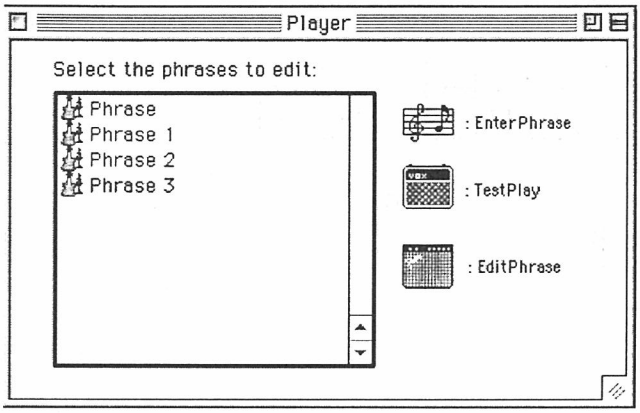


図 2: 演奏家を表すプレイヤー・ウィンドウ。アドリブ・フレーズを貯える。

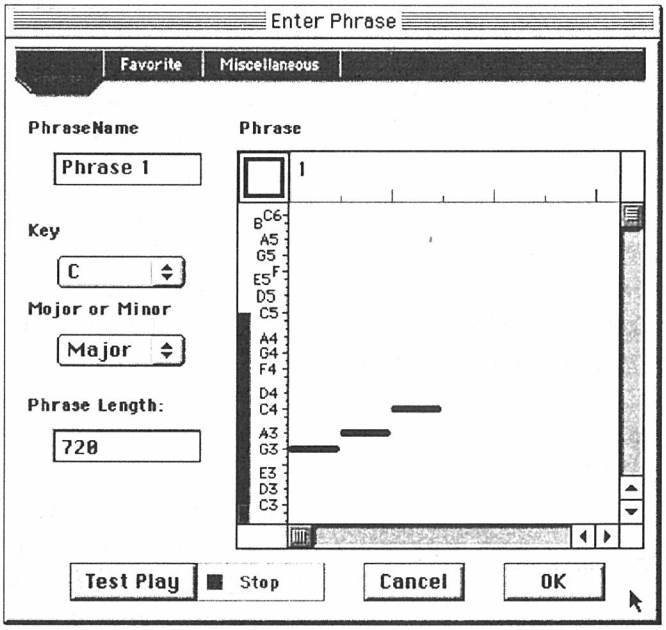


図 3: アドリブ・フレーズの属性値を編集するためのダイアログ

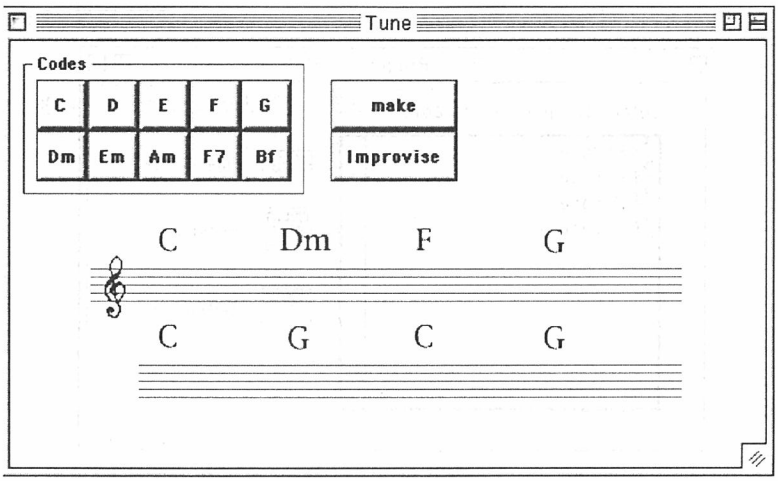


図 4: 楽曲のコード情報を編集、表示するチューン・ウィンドウ

たりする役目を果たす。

アドリブ・フレーズの登録は、第 3.1.2 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、
- オモテ拍で始まる場合とウラ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適、不適。

## 3.3 伴奏としての楽曲部

伴奏となる楽曲の情報を扱う部分で、チューン・ウィンドウを持つ (図 4)。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。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。

## 3.4 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。

# 4 プログラムの実装

## 4.1 開発システム

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。

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。

## 4.2 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から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。アドリブ生成の方法については第2.3章で述べた。

### 4.2.1 OMSとMIDIまわりの実装

本プロジェクト独自に、MIDI関係のクラス群の実装を行なって来たが、これらのクラス群は、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への依存をなるべく小さくするようにした。OMSについては、Opcode社の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を用いることができる。

**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基地クラス」と呼ぶこととする。MIDI基地クラスは、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。MIDI基地クラス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。ただし、PowerPlantの配列クラスが、内部で使用するために、同時に必要である。

**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MIDI基地クラス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、
- 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。OMSのタイマーとは、*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が提供されている。

**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。

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。

**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、コールバック・ルーチン設定、OMSのスタジオ・セットアップ・コマンドなど。

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。

レコーダー・クラスや、レコーディングの際のテンポを一定に保つためのクリック音を生成するクラスは、このクラスを利用することによって、OMS Timingに引き渡す音符シーケンス (フレーズ) の配列を容易に作ることができる。

#### 4.2.2　レコーダー部

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。

#### 4.2.3　アドリブ生成部

第2.3章で述べられたアドリブ生成を実装する部分で、貯えられたアドリブ・フレーズの中からフレーズを選択し、MIDI実装部に引き渡すことによって、リアルタイムにアドリブを生成する。

## 5　結論と今後の課題

実際にアドリブを自動生成させてみると、BoP3はかなり自然な流れを感じさせる演奏を行なう。ただし、今後、他のユーザによる実験を行なって、さらに客観的な評価を行なわなければならない。

BoP3では、ユーザー自らがアドリブ・フレーズを入力できるため、ユーザー自身のアドリブ演奏の再現を行なうことができるということは、ユーザにとって魅力であろう。ただし、現在のところ、創造的で、思いがけない名演奏ができているとまでは言えないかもしれない。アドリブ・フレーズの選択については、さらに高度な手法を開発する必要があると考えられる。また、ユーザーが入力したフレーズだけでなく、新しいフレーズを生み出す方法も開発する必要があると考える。

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ことを考えている。

## 参考文献


[1] Curtis Roads: “The Computer Music Tutorial”, The MIT Press, 1996.

[2] 長島洋一、橋本周司、平賀譲、平田圭二編; “コンピュータと音楽の世界”, bit別冊, 共立出版株式会社, 1998年8月.

[3] 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.

[4] 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.

[5] 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.