私の研究

音楽記述言語「Takt」と生成音楽

西村　憲（にしむら　さとし）
公立大学法人会津大学
情報システム学部門コンピュータ芸術学講座
上級准教授

1．はじめに

本稿では、私がコンピュータグラフィックスについての研究と並行して開発を続けてきた音楽記述言語 Takt[1] を紹介するとともに、それを使った生成音楽について解説したいと思います。コンピュータによる音楽生成、いわゆる自動作曲は、近年の AI ブームの中で注目されている技術の１つであり、作曲家の負担軽減だけでなく、環境に適応して変化する音楽や、作曲の知識を持たない人でもキーワードや簡単なメロディラインを与えるだけで本格的な曲を制作してくれるシステムなどへの応用が期待されています。

音楽記述とは、作曲者から演奏者へ伝えるべき情報を表現したものです。その例としてまず思い浮かぶのは五線譜でしょう。五線譜には音の高低や声部間の関係を把握しやすいという利点があります。しかし、コンピュータで音楽を扱う場合、五線譜では入力に手間がかかり、また高品質の五線譜を出力するには計算コストが大きいという問題があります。そこで、コンピュータ音楽の世界では、五線譜のかわりに文字を使った音楽記述が古くからよく使われてきました。実は、文字を使って音楽を表記するという手法は、コンピュータが世に登場するよりもずっと前から使われており、古くは紀元前後のものとされるセイキロスの墓碑銘に遡りますし、和楽器の楽譜でおなじみの方も多いと思います。

音楽記述言語 Takt も、上で述べたような文字による直接的な表記が基本になっています。しかし、それだけではなく後で述べるように音楽の構造を表現したり、生成音楽の記述、つまり音楽の作り方を表現したりすることができるのが特徴です。生成音楽の記述は、汎用のプログラミング言語によっても可能ですが、開発に手間がかかる上、プログラムを変更してから音が鳴るまでのタイムラグが大きいという問題があります。これに対し、Takt では短い時間でプログラムを作成でき、また、それを入力したら即座に演奏されるような環境が整っています（図１、図２）。なお、Takt 関連のソフトウェアはオープンソースのフリーウェアとして公開されており、だれでも自由に利用することができます[2]。

2．Takt による音楽記述

Takt における文字や記号による音楽表記とは以下のようなものです。例えば、ド、レ、ミという３つの４分音符は ｃｄｅと表されます。４分音符以外は記号をつけて区別し、例えば２分音符の

図１　Takt の起動画面
音楽の記述を打ち込むたびに、その音楽が演奏される。

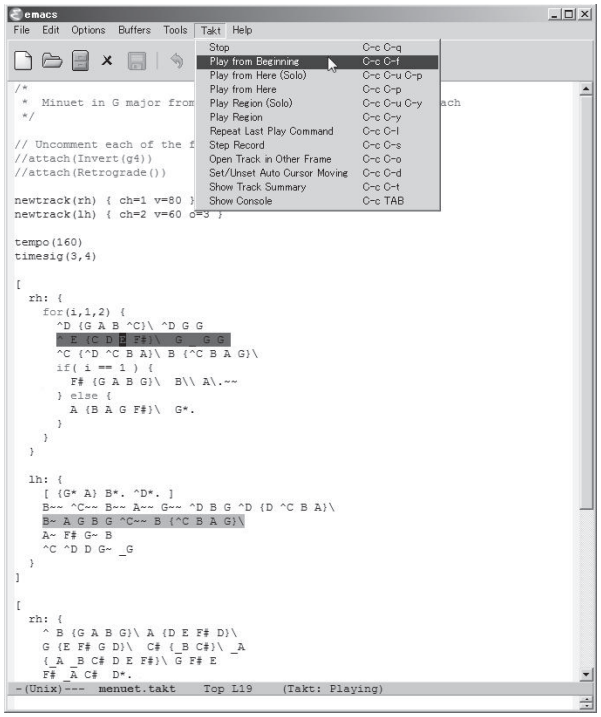

図２　テキストエディタと連携した音楽編集システム
演奏を開始すると現在の演奏位置がハイライトされる。

ドなら c* といった具合です。和音は［c e g］のように角括弧で音符を囲みます。[c*{f e}] のように複数の並行する声部を表すこともできます（この場合 c* が下の声部、f e が上の声部に対応します）。

同じフレーズが何度も出現する場合は、フレーズの定義と呼び出しを利用できます。例えば、

x ={c d e c c d e c e f g* e f g*}

のようにフレーズを x という名前で定義しておくと、以後は x と書くだけで同じフレーズを演奏するようになります（上のフレーズは有名なフレール・ジャックの冒頭部分です）。

フレーズの定義と呼び出しは、音楽の構造を記述するための基本になっています。最も単純な構造の１つである繰り返しは、x x のように単に呼び出しを繰り返すだけで表現できますし、もっと多く繰り返す場合は、repeat(10){x} のような構文も使用できます。

もう少し複雑な構造について見てみましょう。「輪唱」のように時間をずらして同じフレーズを演奏する場合は、少し遅れて演奏する{ r*** x} と元の x の合成として［{ r*** x}x］と書きます。輪唱を発展させたものにカノンがあります。カノンでは、例えば音の高さを５度上げつつ動きを反転させ、さらに時間をずらして演奏するということが行われますが、Takt ではこのような一見複雑な構造も簡単に表現できます。カノンやそれに類似する音楽手法は、例えばバッハのゴールドベルク変奏曲やインベンション（図３）など、実際の曲で活用されています。

上の輪唱の例のような記述には、x がどんなフ

```
var S = {c d e f d e c}¥¥
var sc = Scale.major(c)
var Si = S | Invert(c, sc)

[
  right_hand: {
    r¥¥ S {g ^c b ^c}¥
    ^d¥¥ S|Transpose(4,sc) {^d ^g ^f ^g}¥
    ^e¥¥ Si|Transpose(5+7,sc)
    ^g¥¥ Si|Transpose(3+7,sc)
    ^e¥¥ Si|Transpose(1+7,sc)
  }
  left_hand: {
    r* r¥¥ S|Transpose(-7,sc)
    {_g __g}¥ r r¥¥ S|Transpose(-3,sc)
    {c _b c d e _g _a _b c _e _f# _g}¥
  }
]
```

図３　バッハのハ長調インベンションの構造的な記述（抜粋）
はじめに基本となるフレーズとそれを反転したもの定義し、それらを曲の中で変換をかけながら呼び出している。

レーズであるかという情報は含まれていないことに注目してください。つまり、音楽からその成り立ちだけを抽出したものであると言えます。このような記述をあらかじめ多数用意しておけば作曲のヒントとして役に立ちます。フレーズの方も多数用意しておけば、非常に多くの組み合わせが生まれますから、それらを順に試していき、その中から音楽的に自然なものを選別すれば、思いもかけない良い曲ができあがるかもしれません。さらに、その選別作業をコンピュータに行わせることも徐々に可能になってゆくものと思われます。

3．生成音楽

最も単純な音楽の生成方法は、乱数によって音の高さや音の長さを決めるというものです。いわばサイコロを振ってそれらを決めることに相当します。例えば、下の Takt プログラムは、ピアノのほぼ中央に位置するドの音からその１オクターブ上のドまでの間において（黒鍵の音も含めて）ランダムに選んだ32個の８分音符を発生させます（図４）。

repeat(32){ note(irand(c4, c5))¥}

このような音の列を音楽と呼べるかどうかは議論の分かれるところですが、それでも１つの素材として曲の中で部分的に使用することは十分可能だと思います。

上の例では黒鍵も含めたいわゆる半音階スケールを使用しましたが、たとえば白鍵だけからなるスケールのように、より限定的なスケールを使用するとずっと音楽らしくなってきます（図５）。どんなスケールを使うかは、使用しているコードに合ったものを音楽理論に則して選ぶのが一般的です。また、スケール上の各音は、コードとの親和度にばらつきがありますので、それを考慮して音楽生成を行えばより自然なものが得られます。

生成に使用する数列についても様々なものが考えられています。一様乱数（サイコロを振って得られるような乱数）のかわりに、１/ｆゆらぎに基づいた数列が使われることもよくあり、より音楽らしい音列を発生させることが知られています。また、マルコフ連鎖と呼ばれる１つ前の音をもとに次の音を確率的に決定する手法も使われます。さらに、既存の数列（例えば、円周率の各桁の値、絵画の画像データ、株価データなど）を使う試みも過去に行われてきました。

上では数列によって個々の音符の高さや長さを選んできましたが、かわりにフレーズやリズムパターンを選ぶことも考えられます。つまり、短い音楽要素を自動的に切り貼りしてゆく方法です。この際、前後のつながりが自然になるようにし、

Takt code : repeat(32){ note(irand(c4, c5))¥}

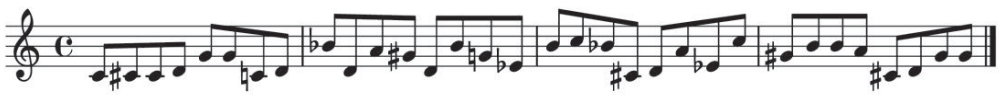

図４　一様乱数によって生成された“音楽”

Takt code : repeat(20){ Scale. major(c4). note(irand(0, 7)) : (l=irand(1, 2)＊1i) }

図５　長音階スケールの音を使ってランダムに生成された音楽
この例では音の長さもランダムに選択している。

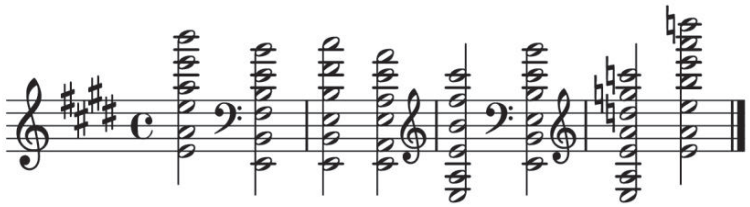

図６　生成された和音の列
完全４度か完全５度をランダムに選んで各音を積み上げている。

また、コードや同時に演奏される他のフレーズとの関係を考慮して選択や修正を行えば、かなり自然な音楽を得ることができます。最近では、歌詞に合うように選ぶようなことも行われています。

その他、数列を使って和音を組み立てるアプローチも面白いです。Takt のサンプルプログラムの中にそのような例が含まれていますが、ベースの音から始めて音程をランダム選んで和音を積み上げてゆくと、時には斬新な響きの和音が得られます（図６）。

近年では、Web の発達にともなって音楽データ（音列データ）が容易に入手できるようになりました（それでも録音データや楽譜画像データと比較するとまだ少ないですが）。それらを音楽生成に役立てる試みも、今後盛んになってゆくものと思われます。既存の曲に含まれる作曲ルールをコンピュータに学習させ、それをもとに上で述べたような手法も駆使しながら音楽を生成させることは可能でしょう。ただし、そのようにして生成された音楽が既存の曲に酷似してしまっては困ります。いくらコンピュータが偶然生成したものだと主張しても、著作権上の問題を生じることは必至です。そこで、そのような可能性を排除するため、大量の曲データを蓄積しておき、それと照合しながら既にある曲を自動的にはじくような仕組みも今後重要になってくるでしょう。

４．おわりに

自動作曲が実用的になるまでにはまだ多くの時間がかかると思われます。１つは技術的な問題から、もう１つは法的な問題からです。似た曲が生成されてしまうという問題とは別に、そもそもコンピュータの作成した音楽に著作権は存在するのか、また存在するとすれば誰に帰属するのかという問題はとても複雑です。しかしいずれは、コンピュータが気の利いた即興演奏者となって我々を常に楽しませてくれるような時代が来ることでしょう。

参考文献

［１］S. Nishimura, “Takt : A read-eval-play-loop interpreter for a structural/procedural score language,” Proc. of the 2014 International Computer Music Conference, 2014.

［２］S. Nishimura, “Takt : Text-based Music Programming Tools,” http://takt. sourceforge. net.

＜プロフィール＞

1965年 埼玉県生まれ。1987年 東北大学工学部卒業、1995年 理学博士（東京大学大学院理学系研究科情報科学専攻）。1994年より会津大学講師、2000年 カリフォルニア大学アーバイン校研究員。2003年より 会津大学助教授（2007年上級准教授に改名）。